UNIVERSITE DE NANCY I

Numéro d'enregistrement au C.N.R.S. :
A.O.10.5 41

T H E S E

présentée à l'Université de Nancy I

pour obtenir le grade de

Docteur-Ingénieur

par

Driss BENLAHCEN

Ingénieur ISEN

"CONTRIBUTION A LA SYNTHESE DES AUTOMATES PAR UN MODULE TETRAEDRIQUE"

Soutenue le 10 Juillet 1975 devant la Commission d'Examen :

Messieurs	F. BOERI	Président
	R. HUSSON	Examinateur
	A. FRUHLING M. LAMOTTE	Rapporteurs

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41547402_0007

UNIVERSITE DE NANCY I

Numéro d'enregistrement au C.N.R.S. :
A.O.10.5 41

T H E S E

présentée à l'Université de Nancy I

pour obtenir le grade de

Docteur-Ingénieur

par

Driss BENLAHCEN

Ingénieur ISEN

"CONTRIBUTION A LA SYNTHESE DES AUTOMATES PAR UN MODULE TETRAEDRIQUE"

Soutenue le 10 Juillet 1975 devant la Commission d'Examen :

Messieurs	F. BOERI	Président
	R. HUSSON	Examinateur
	A. FRUHLING M. LAMOTTE	Rapporteurs

CONTRIBUTION A LA SYNTHESE

DES AUTOMATES PAR UN MODULE TETRAEDRIQUE

UNIV.
LEIDEN
BIBL.

Nous ne pouvons commencer l'exposé de notre mémoire sans remercier tous ceux qui ont contribué à sa réalisation.

Nous remercions vivement Monsieur F. BOERI, Maître de Conférences à l'Université de Nice, qui nous fait l'honneur de présider notre jury de thèse.

Il nous est agréable de remercier Monsieur le Professeur R. HUSSON, pour l'intérêt qu'il a apporté à notre travail et l'accueil qu'il nous a réservé lorsque nous lui avons présenté notre mémoire.

Monsieur le Professeur A. FRUHLING, Directeur du Laboratoire d'Electricité et d'Automatique, en nous accueillant dans son Etablissement, nous a permis d'apprendre ce qu'était le métier de chercheur.

Nous ne saurions exprimer notre gratitude à Monsieur M. LAMOTTE, Chargé de Recherche au CNRS, qui a été tout au long de notre étude un guide et un conseiller attentif.

J'adresse mes remerciements à tous mes camarades du Laboratoire d'Electricité et d'Automatique.

Mesdemoiselles Jocelyne BOGUET et Claudine PRZYBYLSLIK ont contribué à la présentation de ce mémoire ; je les en remercie beaucoup.

TABLE DES MATIERES

	Page
INTRODUCTION	1
CHAPITRE I : Généralités sur les automates	2
I.1 Introduction	2
I.2 Définition	2
I.3 Structure d'un automate	4
I.4 Suite temporelle fondamentale - automate synchrone et asynchrone	4
I.5 Techniques de représentation d'un automate	5
I.6 Etats équivalents	9
I.7 Relation entre les modèles de Moore et de Mealy	11
I.8 Conclusion	13
CHAPITRE II : Décomposition des automates	14
II.1 Intérêt de la décomposition	14
II.2 Interconnexion des automates	14
II.3 Méthodes de décomposition des automates	15
II.3.1. Les partitions	15
II.3.2. Couples de partitions	16
II.3.3. Opérateurs M et m	16
II.3.4. Calcul des différents couples de partitions d'un automate	17
II.3.5. Les recouvrements	27
II.4 Application des partitions et des recouvrements à la décomposition des automates	28
II.4.1. Cas des partitions	28
II.4.1.1. Décomposition série	28
II.4.1.2. Décomposition sans boucle	31
II.4.1.3. Décomposition parallèle	37
II.4.1.4. Décomposition avec boucle	39
II.4.2. Cas des recouvrements	43
II.5 Conclusion	44

Page

CHAPITRE III : Module tétraédrique 45

III.1 Graphe 45

III.1.1. Définition d'un graphe 45

III.1.2. Graphe non orienté 45

III.1.3. Sous-graphe d'un graphe 46

III.2 Automate de Medvedev ou semi-automate 46

III.3 Graphe attaché à un automate 47

III.4 Recherche d'un sous-graphe module 48

III.4.1. Problème à résoudre 48

III.4.2. Dimension du sous-graphe module 48

III.4.3. Sous-graphe module 49

III.5 Module tétraédrique 51

III.5.1. Définition 51

III.5.2. Codage des états du module 51

III.6 Cellule élémentaire associée à un état 52

III.7 Structure de la cellule élémentaire représentant un état 54

III.8 Structure du module tétraédrique 55

III.9 Commandes du module tétraédrique 55

III.10 Fonction de transition du module 58

III.11 Contrainte relative au degré d'un état 58

III.12 Bus de commande x y z h 60

III.13 Schéma logique du module tétraédrique 64

III.14 Conclusion 64

CHAPITRE IV : Application à la synthèse des automates 65

IV.1 But 65

IV.2 Décomposition d'un automate 65

IV.2.1. Méthode 65

IV.2.2. Décomposition non automatique 66

IV.2.3. Recherche des sous-graphes SG 67

IV.2.4. Algorithme de décomposition 68

IV.3 Génération du vecteur de sortie 79

Page

CHAPITRE V : Réalisation - Utilisation 80

V.1 Réalisation 80

V.1.1. Technologie employée 80

V.1.2. Nature des signaux 80

V.1.3. Fonctionnement du module 81

V.1.3.1. Transitions entre états d'un même module 81

V.1.3.2. Diagramme temporel des transitions à l'intérieur d'un même module et entre modules 84

V.1.3.3. Commandes et réponse du module 87

V.1.3.4. Forme d'onde des différents signaux 90

V.1.3.5. Adaptateur de commande 90

V.1.3.6. Calibrage des impulsions données par l'adaptateur 92

V.1.3.7. Entrées-sorties de l'adaptateur de commandes 92

V.1.3.8. Simulateur 92

V.2 Utilisation 93

CONCLUSION 95

BIBLIOGRAPHIE 114

INTRODUCTION

Lors de la conception d'un système séquentiel complexe, on se heurte à deux problèmes principaux :

1) la dimension du système,
2) le codage des états.

La solution du premier consiste à décomposer le système en sous-automates plus petits. La méthode de Hartmanis J. et Stearns R.-E. [1] basée sur les partitions peut, dans certains cas, résoudre ce problème.

Quant au second, il est résolu par Huffman, dont toutefois la méthode est laborieuse. Girard P. et Naslin P. [2] ont introduit la notion d'étapes pour simplifier la synthèse des automates. Pour surmonter le problème de codage, la tendance est de concevoir une méthode de synthèse qui lui soit indépendante. Ce changement d'optique est possible grâce à la technologie des circuits intégrés. Des surfaces cellulaires ou réseaux ont été élaborées pour les circuits combinatoires et séquentiels. En France, dans le domaine séquentiel, la synthèse cellulaire des automates a été développée par R. David [3] , avec la cellule universelle pour séquences asynchrones ou CUSA.

L'objet de ce travail est la conception d'un module tétraédrique qui apporte une contribution à la synthèse simple des automates.

CHAPITRE I

GENERALITES SUR LES AUTOMATES

I-1 INTRODUCTION.

La théorie des automates peut être considérée comme étant l'étude du comportement dynamique des systèmes d'information à paramètres discrets. Elle est basée sur le fait que le comportement extérieur d'un système physique peut être décrit par un modèle mathématique.

I-2 DEFINITION. [1]

Un automate est un quintuplet :

$$A = (X, Y, Z, f, g)$$

où a) X est un ensemble non vide d'entrées ou alphabet,
b) Y est un ensemble non vide d'états internes,
c) Z est un ensemble non vide de sorties,
d) $f:Y.X \rightarrow Y$ est la fonction état suivant,
e) $g:Y.X \rightarrow Z$ est la fonction de sortie.

Les ensembles X, Y, Z ont une coordinalité finie ; certains auteurs [4] appellent un automate une machine à états finis.

Suivant la nature de l'application g on distingue deux sortes d'automates :

- modèle de Moore :
  La fonction g est indépendante de X :
  $$Z = g(Y)$$

- modèle de Mealy :

g est une fonction de X et Y :

$$Z = g(Y,X).$$

Monoïde X^* :

Une suite finie α d'éléments de X est un mot sur X :

$$\alpha = x_1 \, x_2 \, \ldots \, x_n$$

n est la longueur du mot.

L'ensemble des mots, y compris le mot vide, muni de la loi de composition interne associative, la concaténation ou juxtaposition et admettant comme élément neutre le mot vide, constitue le monoïde X^*.

Un langage sur l'alphabet d'entrée X est un sous-ensemble de X^*.

L'extension des applications f et g aux monoïdes X^* et Y^* définis sur les alphabets X et Y est déterminée par :

$$f(y,xx') = f(f(y,x),x')$$
$$g(y,xx') = g(g(y,x),x').$$

Les fonctions f et g ne sont pas toujours définies. On note par exemple

$$g(y,x) = \emptyset.$$

C'est ce qui se passe lorsque, dans un automate, une situation ne se rencontre pas. Par exemple, à une combinaison impossible du vecteur d'entrée ne correspond aucune sortie.

I-3 STRUCTURE D'UN AUTOMATE.

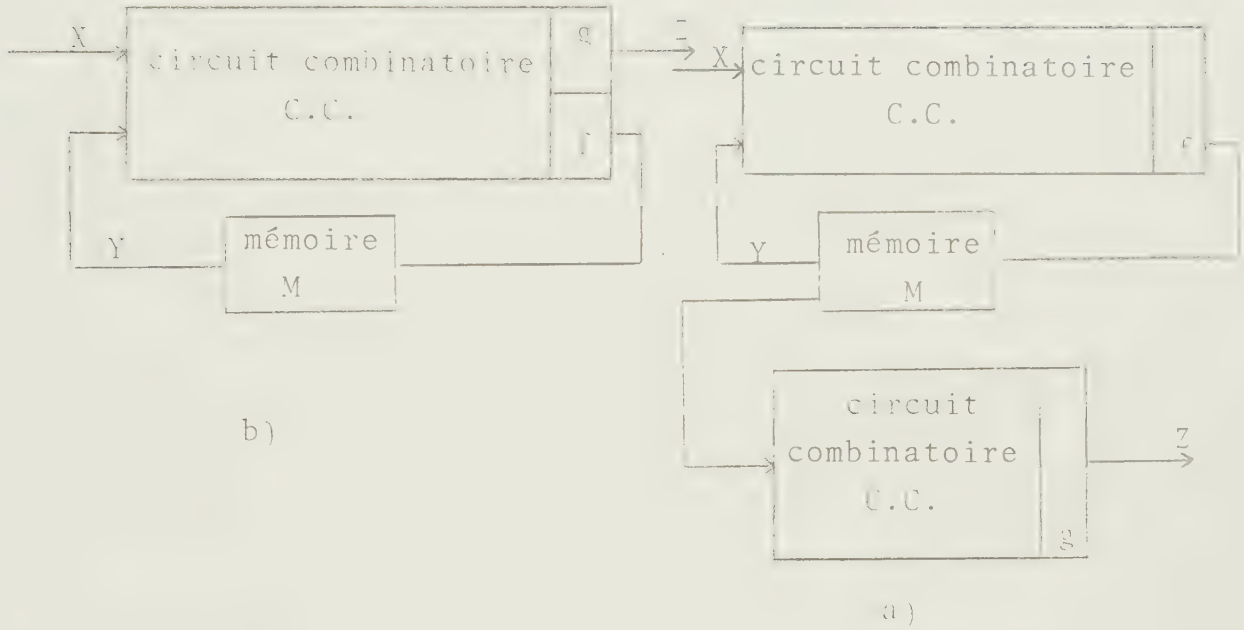


Fig. 1.1

La figure 1.1 montre les deux modèles d'un automate :

- le schéma a) représente le modèle de Moore,
- le schéma b) représente le modèle de Mealy ou d'Huffmann.

Chaque modèle comprend un bloc combinatoire qui calcule les fonctions f et g, avec une mémoire qui stocke les informations nécessaires à ce calcul. L'ensemble Y des informations contenues dans la mémoire est l'ensemble des états internes Y du système. L'état total de l'automate est le couple $(x,y) \in (X.Y)$.

I-4 SUITE TEMPORELLE FONDAMENTALE. AUTOMATES SYNCHRONE ET ASYNCHRONE.

On considère une suite d'instants successifs que l'on peut repérer par 1, 2, 3... et que l'on appelle "suite temporelle fondamentale". Ces instants sont en général concré-

tisés par des signaux d'horloge qui se présentent comme une suite d'impulsions.

On distingue deux sortes d'automates :

l'automate synchrone dont la réponse à une commande nécessite un signal d'horloge,

l'automate asynchrone qui évolue librement dès qu'une commande est appliquée.

Dans les deux cas, le calcul des fonctions f et g nécessite un temps τ minimum qui est le temps de réponse du système. Deux instants successifs de la suite temporelle fondamentale doivent être distants d'au moins τ , afin d'éviter des interférences entre deux commandes successives.

Les équations de récurrence qui décrivent l'automate sont :

$$Y_{n+1} = f(Y_n, X_n)$$
$$Z_n = g(Y_n, X_n).$$

I-5 TECHNIQUES DE REPRESENTATION D'UN AUTOMATE.

a) Table des transitions :

C'est un tableau dont les colonnes correspondent aux différentes entrées et les lignes aux différents états de l'automate. L'intersection de la colonne x et de ligne y contient dans l'ordre f(y,x) et g(y,x).

b) Matrice des transitions :

Si p est le nombre des états de l'automate, la matrice de transition est une matrice p.p dont les colonnes représentent les états suivants et les lignes les états présents. L'intersection d'une ligne et d'une colonne contient

les entrées x_{ij} qui permettent de passer de l'état i à l'état j et la sortie correspondant à cette transition.

c) Graphe de transition ou diagramme de transition :

L'ensemble des sommets du graphe correspond aux états de l'automate. A tout couple ordonné (y_i, y_j) correspond un arc orienté de y_i vers y_j s'il existe des entrées x_k telles que $f(y_i, x_k) = y_j$. L'arc est marqué par $x_k/g(y_i, x_k)$. On peut avoir plusieurs x_k : x_{k_1}, ... x_{k_m}, qui font passer d'un état y_i au même état y_j.

Exemple (figure 1.2) : Considérons un automate à 4 états, ayant une seule sortie et commandé par une variable. La sortie et les transitions d'état à état dues au changement de la commande sont :

Commande	Transition d'état à état		Sortie
0	état 1	à état 1	0
1	1	3	0
1	2	2	1
0	2	4	1
0	3	2	0
1	3	4	1
0	4	1	1
1	4	3	0

Fig. 1.2

Pour cet automate, la table de transition :

états \ entrée	0	1
1	1/0	3/0
2	4/1	2/1
3	2/0	4/1
4	1/1	3/0

Fig. 1.3

et la matrice des transitions sont représentées sur les fig. 1.3 et 1.4

états présents \ états suivants	1	2	3	4
1	0/0	-	1/0	-
2	-	1/1	-	0/1
3	-	0/0	-	1/1
4	0/1	-	1/0	-

Fig. 1.4
Matrice des transitions

Dans ces représentations, une case vide indique un manque de commande et de transition à cet emplacement.

Le graphe de transitions est représenté sur la Fig. 1.5

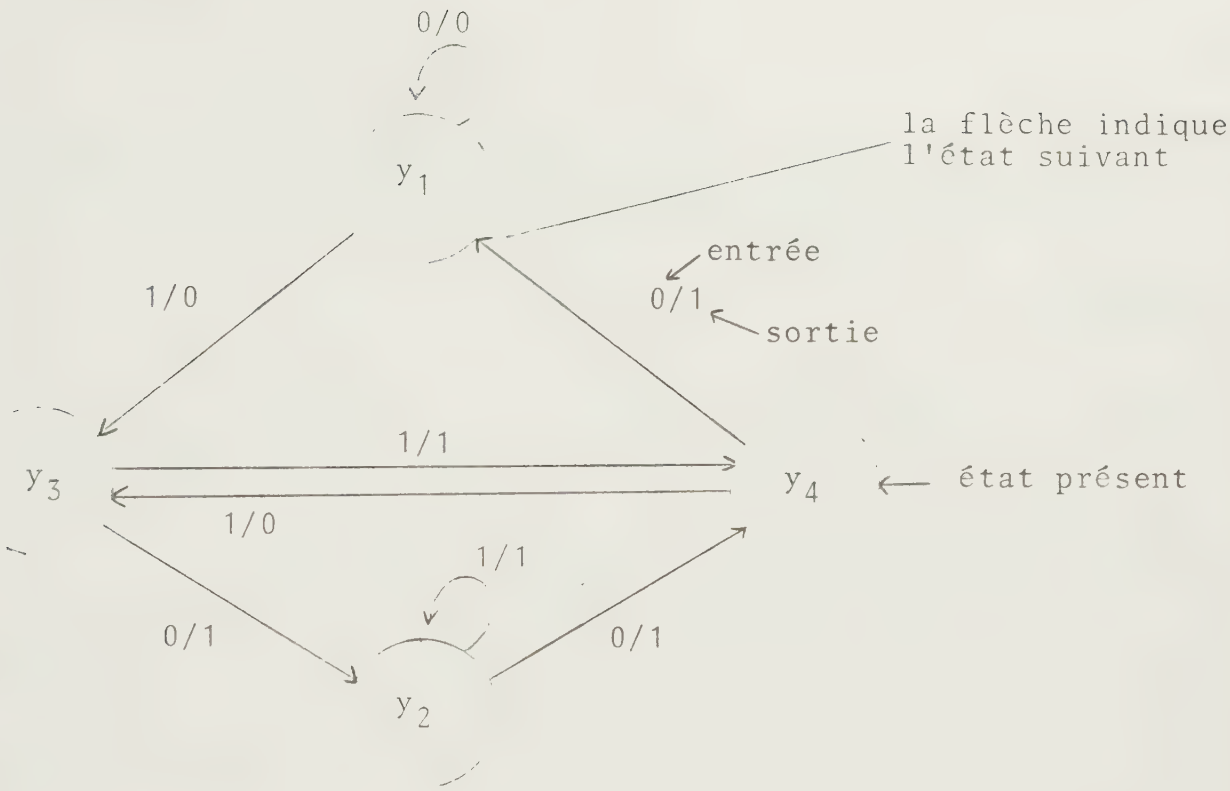


Fig. 1.5

Diagramme de transitions pour un modèle de Mealy.

d) Expressions régulières :

On a vu au paragraphe I-2 que l'on peut définir les fonctions f et g sur les monoïdes X^* et Y^*. Ainsi, à toute séquence d'entrées correspond une séquence de sorties. On appelle langage défini sur X, alphabet d'entrée, toute partie de X^*.

Soit un automate $A = (X,Y,\Sigma,f,g)$; le langage accepté par A (on dit aussi reconnu) est la partie de X^* pour laquelle l'automate A est défini.

Sur l'ensemble X^*, on définit deux lois de compositions internes, l'addition et l'itération notées respectivement + et $*$.

L'ensemble X^* muni des deux lois + et * constitue l'ensemble des expressions régulières.

Exemple :

$$R_1 = x_1x_2 + x_3$$

$$R_2 = x_1x_2(x_3x_1)^* = x_1x_2x_3x_1+x_1x_2x_3x_1x_3x_1+x_1x_2x_3x_1x_3x_1+.$$

Exemple d'automate défini par les expressions régulières :

Soit un automate ayant 4 vecteurs d'entrée E_1, E_2,E_3,E_4 et 3 vecteurs de sortie Z_1,Z_2,Z_3, tel que, à la séquence d'entrée $E_1E_2E_4E_3(E_1E_2)^*$, correspond en sortie la séquence suivante :

$$Z_1Z_2Z_2Z_1(Z_3Z_1)^*.$$

A la séquence d'entrée $E_1E_3(E_1E_3)^*$ correspond la séquence de sortie $Z_1Z_1(Z_3Z_1)^*$.

Les expressions régulières qui définissent l'automate sont :

$$R(Z_1) = E_1+E_1E_2E_4E_3 + E_1E_3 + E_1E_2E_4E_3(E_1E_2)^* + E_1E_3(E_1E_3)^*$$

$$R(Z_2) = E_1E_2 + E_1E_2E_4$$

$$R(Z_3) = E_1E_2E_4E_3E_1(E_2E_1)^* + E_1E_3E_1(E_2E_1)^*.$$

Les expressions régulières trouvent leur emploi dans la synthèse des automates par la méthode de Glushkov [5] .

I-6 ETATS EQUIVALENTS-REDUCTION D'UN AUTOMATE.

Soient L_k l'ensemble des mots de longueur inférieure ou égale à k et L_∞ l'ensemble de tous les mots de longueur finie ; L_k et L_∞ sont définis sur l'alphabet d'entrée X de l'automate.

k-équivalence et équivalence de deux états :

Deux états y_i et y_j sont k-équivalents si l'on a :

(1) $g(y_i,l) = g(y_j,l) \quad \forall \; l \in L_k$.

Dans le cas où la relation (1) n'est pas vérifiée, on dit que y_i et y_j sont k-distinguables.

Si la relation (1) est vraie pour tout $l \in L_\infty$, on dit que y_i et y_j sont équivalents. La relation (1) induit une partition Π sur l'ensemble Y des états.

Forme réduite d'un automate :

Soient $C_1, C_2, \ldots C_m$ les classes d'équivalence de l'automate A. Définissons l'automate $A_r = (X, Y_r, Z, f_r, g_r)$ de la façon suivante :

$$Y_r = (y_{r1}, y_{r2}, \ldots, y_{rm})$$

Si $f(y_i,x) \in C_k$ alors $f_r(y_{ri},x) = y_{rk}$

Si $g(y_i,x) = z_k$ alors $g_r(y_{ri},x) = z_k$.

L'automate A_r est la forme réduite de l'automate A.

Exemple : soit l'automate A défini par la table de transition Fig. 1.6

entrée / états	E_1	E_2
1	3/0	5/0
2	1/0	6/0
3	3/0	5/1
4	4/0	6/1
5	5/0	1/0
6	6/0	2/0

Fig. 1.6

La partition Π obtenue pour cet automate est :

$$\Pi = (12,\ 34,\ 56).$$

Le nombre des états de l'automate réduit A_r est égal au nombre des blocs de Π . En supprimant les états redondants et en numérotant de nouveau les états, on obtient la table réduite de la fig. 1.7 :

entrée / états	E_1	E_2
1	2/0	3/0
2	2/0	3/0
3	3/0	1/0

Fig. 1.7

I-7 RELATION ENTRE LES MODELES DE MOORE ET MEALY.

Les deux modèles d'automates définis au paragraphe I-3 diffèrent dans la manière de générer la sortie. Dans le cas d'un automate de Moore, la sortie qui dépend seulement de l'état est toujours disponible, sauf durant une transition ; l'automate de Mealy au contraire ne peut avoir une sortie que lorsqu'une entrée est appliquée.

Si les observations du comportement d'un système séquentiel ne se font qu'aux instants déterminés par la suite temporelle fondamentale, les modèles de Moore ou de Mealy peuvent être utilisés indifféremment pour représenter le système soumis à l'étude.

Transformation d'un automate de Moore en un automate de Mealy :

Soit MO = (X,Y,Z,f,g) un automate de Moore. On veut définir un automate ME = (X',Y',Z',f',g') de Mealy qui doit avoir les mêmes entrées-sorties que MO.

L'état initial doit être ignoré quand on compare un modèle de Moore à un modèle de Mealy.

La correspondance entre les deux automates est donnée par les relations suivantes :

a) $X' = X$; $Y' = Y$; $Z' = Z$

b) $f'(y_i,x_j) = f(y_i,x_j)$

c) $g'(y_i,x_j) = z_j$ si $f(y_i,x_j) = y_k$ et $g(y_k) = z_j$.

La figure I.8 montre l'équivalence des diagrammes de transition des deux modèles.

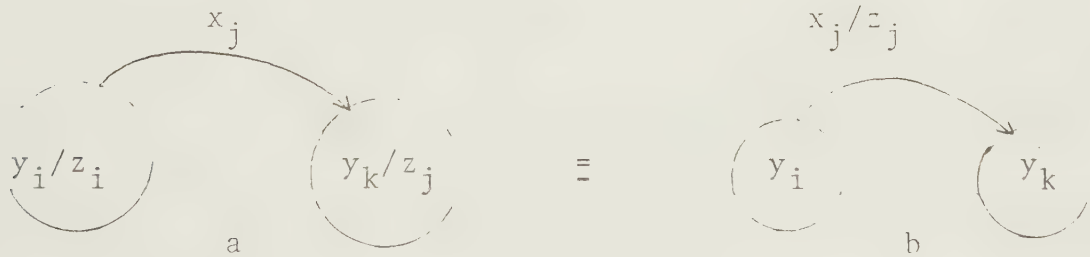


Fig. 1.8

a-Modèle de Moore ; b-son équivalent de Mealy

Conversion d'un modèle de Mealy en un modèle de Moore :

Soit ME(X,Y,Z,f,g) un automate de Mealy. On veut définir un automate de Moore MO = (X',y',Z',f',g') qui doit avoir les mêmes entrées-sorties que ME.

La correspondance se traduit par :

a) $X' = X$, $Z' = Z$

b) $Y' = \left\{ \begin{array}{l} y'_{ki} = (y_k, z_i) \text{ tel que : } \exists\, x_j \in X \begin{cases} f(y_j, x_j) = y_k \\ g(y_j, x_j) = z_i \end{cases} \\ \qquad\qquad\qquad\qquad\qquad\qquad \exists\, y_j \in Y \\ \text{ou} \\ y'_{k,-} = y_k \text{ : s'il n'existe aucune séquence } J \\ \qquad\qquad \text{telle que } f(y_k, J) = y_k \end{array} \right\}$

c) $f'(y'_{ki}, x_1) = y'_{mn} = (f(y_k, x_1) = y_m \;,\; g(y_k, x_1) = z_n)$

d) $g'(y'_{ki}, x_1) = z_i$ ou $g'(y'_{ki}, x_1) = \emptyset$ si la séquence J n'existe pas.

Exemple : La figure 1.9 montre les changements nécessaires pour passer d'un automate de Mealy à un automate de Moore.

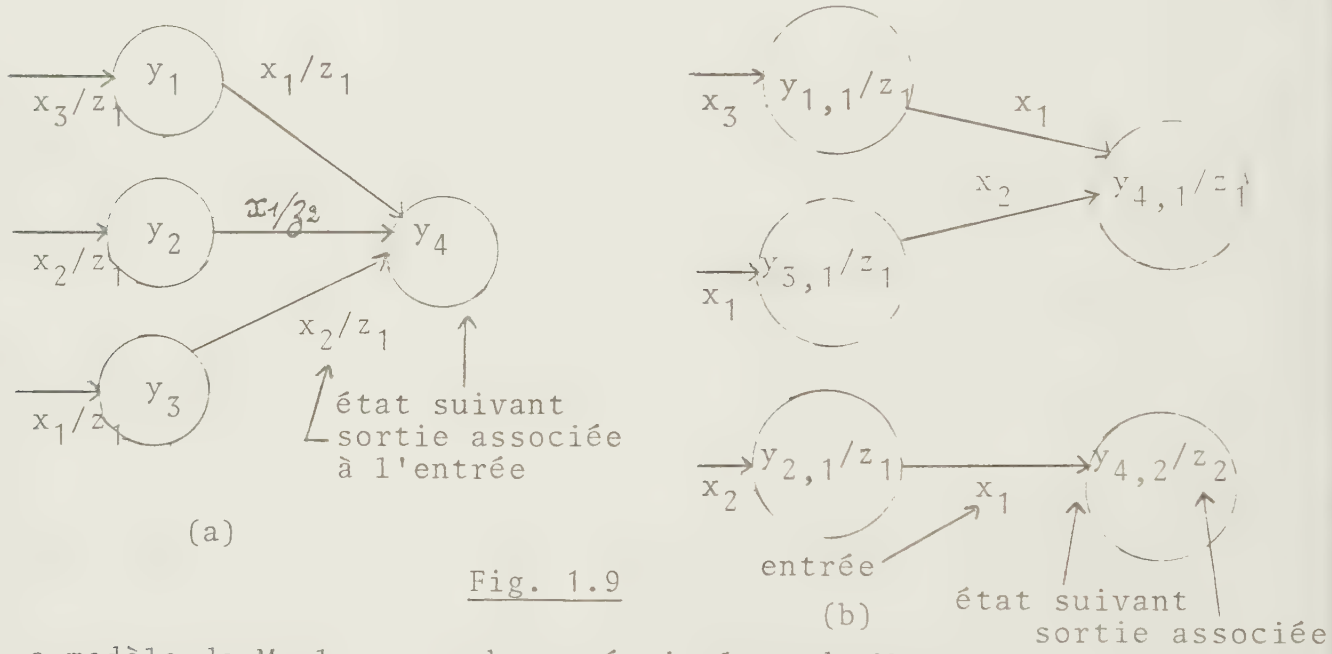


Fig. 1.9

a-modèle de Mealy ; b-son équivalent de Moore.

I-8 CONCLUSION.

On a rappelé dans ce chapitre les trois méthodes de représentation d'un automate. La table de transitions est commode pour l'étude des automates à l'aide des partitions. La représentation graphique, plus parlante, donne une idée de la complexité de l'automate relatif au graphe. Qu'il s'agisse d'un automate de Moore ou de Mealy, on utilisera principalement cette méthode pour son étude.

CHAPITRE II
DECOMPOSITION DES AUTOMATES

L'objet de ce chapitre est de montrer les différentes techniques utilisées pour décomposer un automate. Les concepts présentés sont dûs principalement à Hartmanis et Stearns [1] .

II-1 INTERET DE LA DECOMPOSITION.

Pour simplifier sa construction, son installation, sa maintenance, il est souhaitable qu'un automate soit formé d'automates moins importants. L'intégration de la fabrication d'un automate à moyenne échelle peut être élaborée si celui-ci n'est pas trop gros ou s'il est décomposable en sous-automates intégrables.

II-2 INTERCONNEXION DES AUTOMATES.

La composition d'un automate au moyen de sous-automates peut se réduire à trois formes principales, comme le montre la figure 2.1 où le bloc c.c désigne un circuit combinatoire.

a) Composition parallèle.
b) Composition série ou cascade.
c) Composition avec boucle.

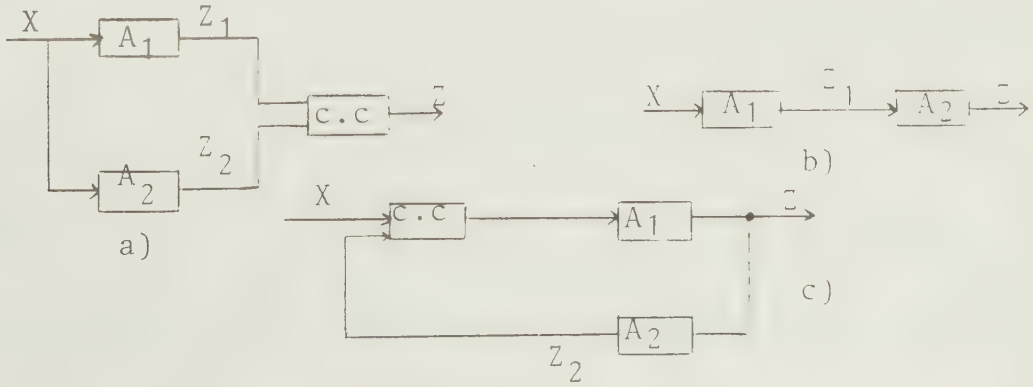


Fig. 2.1
Principales formes de composition

Les combinaisons de ces trois formes principales sont évidemment possibles entre elles.

II-3 METHODES DE DECOMPOSITION DES AUTOMATES.

Les couples de partitions sont utilisés pour la décomposition d'un automate.

II.3.1. Les partitions.

Définition : Une partition π d'un ensemble Y est une collection de sous-ensembles B_i de Y vérifiant les conditions suivantes :

$$\pi = \{B_i\}$$
$$B_i . B_j = \emptyset \quad \text{pour } i \neq j$$
$$\sum_i B_i = Y.$$

Les éléments B_i de π sont les blocs de la partition.

Relation d'ordre partielle :

Soient $\pi_1 = \{B_{1i}\}$ et $\pi_2 = \{B_{2j}\}$ deux partitions définies sur le même ensemble Y. On dit que :

$$\pi_1 \leq \pi_2 \iff \forall B_{1i} \exists B_{2j} : B_{1i} \leq B_{2j}$$

$$\pi_1 \leq \pi_2 \iff \pi_1 . \pi_2 = \pi_1 \text{ et } \pi_1 + \pi_2 = \pi_2.$$

Partitions triviales :

$I = \{Y\}$: partition formée d'un seul bloc, l'ensemble Y

$O = \{y_1, y_2, \ldots, y_n\}$: chaque bloc est un élément de Y et $|Y| = n$.

Propriété : L'ensemble des partitions d'un ensemble forme un treillis

II.3.2. Couple de partitions.

Définition : Pour l'automate $A = \{X,Y,Z,f,g\}$, un couple de partition (π,π') défini sur A est un couple ordonné de partitions π et π' définies sur Y telles que si s et t sont deux éléments d'un même bloc de π, leurs transformées par la fonction f appartiennent à un même bloc de la partition π' pour toute entrée x. Cette définition est traduite symboliquement par :

$$s \equiv t(\pi) \Rightarrow f(s,x) \equiv f(t,x) \ (\pi') \quad \forall\, x \in X.$$

Partition avec propriété de substitution :

Une partition π est une partition à propriété de substitution si et seulement si $(\pi\ ,\pi\ )$ est un couple de partitions défini sur A.

Relation d'ordre partielle :

Soient deux couples (π_1, π_2) et (π'_1, π'_2) définis sur un automate ; on définit l'inégalité de deux couples par :

$$(\pi_1,\pi_2) \leqslant (\pi'_1,\pi'_2) \iff \begin{array}{l} \pi_1 \leqslant \pi'_1 \\ \pi_2 \leqslant \pi'_2 \end{array} .$$

II.3.3. Opérateurs M et m.

Pour une partition π d'un ensemble Y, on définit :

$$m(\pi) = \Pi\{\pi_i \mid (\pi,\pi_i) \quad \text{est un couple de partitions de } Y\}$$

$$M(\pi) = \Sigma\{\pi_i \mid (\pi_i,\pi) \quad \text{est un couple de partitions de } Y\} .$$

Couple Mm :

Un couple de partitions (π, π') est un couple Mm si $\pi = M(\pi')$ et $\pi' = m(\pi)$.

II.3.4. *Calcul des différents couples de partitions d'un automate.*

1) *Calcul de l'ensemble CP des couples de partitions d'un automate :*

a) Propriétés des couples de partitions :

Soient deux couples de partitions (π_1,π_2) et (π'_1,π'_2) tels que :

$$\pi'_1 \leqslant \pi_1 \quad \text{et} \quad \pi_2 \leqslant \pi'_2 .$$

On a alors deux autres couples de partitions :

$$(\pi'_1,\pi_2) \quad \text{et} \quad (\pi_1,\pi'_2) .$$

De même, en considérant le plus grand minorant et le plus petit majorant des couples (π_1,π_2) et (π'_1,π'_2), on obtient deux autres couples de partitions :

$$(\pi_1 \cdot \pi'_1 \ , \ \pi_2 \cdot \pi'_2) \quad \text{et} \quad (\pi_1 + \pi'_1 \ , \ \pi_2 + \pi'_2).$$

Si on peut trouver un couple de partitions (π_1,π_2) :

- toute partition π'_i telle que :

$$\pi'_i \leqslant \pi_1$$

permet de générer la partition (π'_i,π_2) ;

- toute partition π'_j telle que :

$$\pi_2 \leqslant \pi'_j$$

donne le couple (π_1,π'_j).

- A toute partition π , on peut associer les partitions triviales I et O pour former deux couples de partitions :

$$(\pi,I) \quad , \quad (O,\pi) .$$

b) Partitions π_{ij} et π^s_{ij} :

- Les blocs de la partition π_{ij} se réduisent à un seul élément, à l'exception de l'un d'entre eux qui est formé de i et j.

- π^s_{ij} est la plus petite partition possédant la propriété de substitution et dans laquelle i et j sont dans un même bloc.

c) Calcul de CP :

On peut donc déterminer l'ensemble CP des couples de partitions d'un automate en mettant en œuvre les résultats précédents.

Exemple :

Soit un automate $A = (X,Y,Z,f,g)$ défini par la table des transitions de la figure 2.1.

$X = \{x_1,x_2,x_3,x_4\}$

$Y = \{1,2,3,4\}$

	x_1	x_2	x_3	x_4
1	1	2	3	4
2	3	4	1	2
3	2	1	4	3
4	4	3	2	1

Fig. 2.1

En application de la définition du couple (π,π') :

$$i \equiv j(\pi) \Rightarrow f(i,x) \equiv f(j,x) \ (\pi') \quad \forall\, x \in X .$$

On détermine toutes les partitions π de l'ensemble Y des états de l'automate et leurs transformées par l'application f pour obtenir les couples (π,π'). A partir de l'ensemble de tous les couples précédents, on considère les minorants et les majorants des paires des couples (π_i,π_j) et (π_k,π_l) qui donnent des couples de partitions supplémentaires. L'ensemble CP est formé de tous les couples de partitions.

On considère toutes les partitions π_{ij} contenant un seul bloc à deux éléments, les autres blocs étant formés d'un seul élément et leurs transformées π'_{ij}.

$\pi_{12} = (12,3,4)$: $f(\pi_{12},x) = \pi'_{12} = (13,24)$

$\pi_{13} = (13,2,4)$: $f(\pi_{13},x) = \pi'_{13} = (12,34)$

$\pi_{14} = (14,2,3)$: $f(\pi_{14},x) = \pi'_{14} = (14,32)$

$\pi_{23} = (23,1,4)$: $f(\pi_{23},x) = \pi'_{23} = (32,14)$

$\pi_{24} = (24,1,3)$: $f(\pi_{24},x) = \pi'_{24} = (34,12)$

$\pi_{34} = (34,1,2)$: $f(\pi_{34},x) = \pi'_{34} = (24,13)$.

La somme de deux partitions $\pi_{ij} + \pi_{kl}$ est la partition triviale I. De même, le produit de deux partitions $\pi'_{ij}.\pi'_{kl}$ est la partition triviale O. Et comme toute partition π peut former avec O et I les couples de partitions (π,I) et (O,π), ces couples n'apportent aucune nouvelle information concernant le comportement de l'automate en plus de celle contenue dans la partition π.

Pour l'automate de la figure 2.1, l'ensemble des couples de partitions CP est :

$$CP = \{(\pi_{ij} , \pi'_{ij})\}$$

où π_{ij} et π'_{ij} sont les partitions considérées auparavant.

2) *Calcul de l'ensemble CM des couples de partitions Mm :*

Le calcul des couples Mm est justifié par le théorème de Hartmanis [1] :

si (π,π') est un couple Mm, on a :

$$\pi' = \Sigma \{m(\pi_{ij}) \mid \pi_{ij} \leq \pi\}$$

$$\pi = \Sigma \pi_{ij}$$

En effet :

$$m(\pi_{ij}) = \Pi\{\pi_k \mid (\pi_{ij},\pi_k) \text{ est un couple de partitions}\}$$

Comme par hypothèse $\pi_{ij} \leqslant \pi$, en faisant les produits des partitions des couples (π,π') et (π_{ij},I), on obtient le couple $(\pi.\pi_{ij}\ ,\ \pi'I) = (\pi_{ij},\pi')$ puisque $\pi_{ij} \leqslant \pi$ et $\pi' \leqslant I$.

Or, $m(\pi_{ij}) = \Pi\{\ \pi_k \mid (\pi_{ij},\pi_k)$ est un couple de partitions$\}$ ce qui implique que π' est l'une des $\pi_k \Rightarrow \pi' \geqslant m(\pi_{ij})$.

Considérons les deux inégalités suivantes :

$$m(\pi_{ij}) \leqslant \pi'$$

$$\pi_{ij} \leqslant \pi\ .$$

Faisons la somme de toutes les inégalités relatives aux π_{ij} :

$$\begin{cases} \Sigma\ m(\pi_{ij}) \leqslant \pi' \\ \Sigma\ \pi_{ij} = \pi \end{cases} \qquad (1)\ .$$

Par définition de $m(\pi_{ij})$, $(\pi_{ij},\ m(\pi_{ij})$ est un couple de partitions ; or, si (π_1,π_2) et (π_1',π_2') sont deux couples de partitions, $(\pi_1 + \pi_1'\ ,\ \pi_2 + \pi_2')$ est aussi un couple de partitions.

Considérons tous les couples $(\pi_{ij},\ m(\pi_{ij}))$ avec $\pi_{ij} \leqslant \pi$ et faisons leur somme ; on obtient le couple de partitions :

$$(\Sigma\ \pi_j\ ,\ \Sigma\ m(\pi_{ij})) = (\ \pi\ ,\ \Sigma\ m(\pi_{ij}))\ .$$

Si (π,π') est un couple Mm, on a par définition :

$$\pi' = m(\pi) = \Pi \{\pi_1 | (\pi,\pi_1) \text{ est un couple de partitions}\}$$

donc, parmi toutes les partitions π_1, on trouve $\Sigma\, m(\pi_{ij})$

$$\pi' \leqslant \pi_1 \quad \forall\, \pi_1 \text{ tel que } (\pi,\pi_1) \text{ est un couple de partitions,}$$

$$\Rightarrow \quad \pi' \leqslant \Sigma\, m(\pi_{ij}) \qquad (2).$$

Des inégalités (1) et (2) on tire :

$$\pi' = \Sigma\; m(\pi_{ij}).$$

De même si (π,π') est un couple Mm, on a par définition :

$$\pi = M(\pi') = \Sigma\, [\pi_{ij} | (\pi_{ij},\pi')] .$$

Connaissant π', on déterminera π par une sommation des π_{ij}, telles que $m(\pi_{ij}) \leqslant \pi'$.

Le résultat précédent est utilisé pour le calcul des couples Mm :

1) On cherche toutes les partitions $m(\pi_{ij})$ distinctes, ce qui nécessite au plus $\frac{1}{2}\, p(p-1)$ opérations pour un automate à p états.

2) On forme toutes les sommes possibles $\Sigma\, m(\pi_{ij})$ distinctes, ce qui donne tous les éléments m de l'automate.

3) Les partitions homologues M s'obtiennent, pour les partitions m, de la façon suivante :

π'_j est une partition m générée à l'étape 1) ou 2).

La partition $\pi_j = M(\pi'_j) = \Sigma\ (\pi_{ij} \mid m(\pi_{ij}) \leq \pi'_j)$

$CM = \{(\pi_j\ ,\ \pi'_j)\}$.

Exemple de calcul de l'ensemble CM :

La figure 2.2 donne la table de transitions d'un automate :

	x_1	x_2	x_3	x_4
1	5	4	5	2
2	4	1	1	2
3	5	1	3	4
4	4	4	4	2
5	4	4	2	4

Fig. 2.2.

L'examen du tableau donne pour les différentes partitions :

$m(\pi_{12}) = (145\ ,\ 2\ ,\ 3) = \pi'_1$

$m(\pi_{13}) = (124\ ,\ 35) = \pi'_2$

$m(\pi_{14}) = (1\ ,\ 2\ ,\ 3,45) = \pi'_3$

$m(\pi_{15}) = (1\ ,\ 245\ ,\ 3) = \pi'_4$

$m(\pi_{23}) = (13\ ,\ 245) = \pi'_5$

$m(\pi_{24}) = (14,\ 2,\ 3,\ 5) = \pi'_6$

$m(\pi_{25}) = (124,\ 3,\ 5) = \pi'_7$

$m(\pi_{34}) = (1\ 2\ 3\ 4\ 5) = I$

$m(\pi_{35}) = (145\ ,\ 23) = \pi'_8$

$m(\pi_{45}) = (1,\ 24,\ 3,\ 5) = \pi'_9$

Les partitions m, autres que la partition O, s'obtiennent en considérant toutes les sommes distinctes $\Sigma\, m(\pi_{ij})$. La nouvelle partition obtenue est :

$$\pi'_3 + \pi'_7 = (1\ 2\ 4\ 5\ ,\ 3) = \pi'_{10}\ .$$

En utilisant les partitions m précédentes, on détermine les partitions M correspondantes :

$$M(\pi'_i) = \Sigma\ (\pi_{ij} \mid m(\pi_{ij}) \leqslant \pi'_i\ )$$

$$M(\pi'_1) = \pi_{12} + \pi_{14} + \pi_{24} = (124\ ,\ 3\ ,\ 5)$$

car :

$$m(\pi_{12}) \leqslant \pi'_1$$

$$m(\pi_{14}) \leqslant \pi'_1$$

$$m(\pi_{24}) \leqslant \pi'_1\ .$$

De la même manière, on trouve :

$$\pi_2 = M(\pi'_2) = \pi_{13} + \pi_{24} + \pi_{25} + \pi_{45} = (245\ ,\ 13)$$

$$\pi_3 = M(\pi'_3) = \pi_{14} = (14\ ,\ 2\ ,\ 3\ ,\ 5)$$

$$\pi_4 = M(\pi'_4) = \pi_{14} + \pi_{15} + \pi_{45} = (145\ ,\ 2\ ,\ 3)$$

$$\pi_5 = \pi_{14} + \pi_{15} + \pi_{23} + \pi_{45} = (145\ ,\ 23)$$

$$\pi_6 = (24\ ,\ 1\ ,\ 3\ ,\ 5)$$

$$\pi_7 = (245\ ,\ 1\ ,\ 3)$$

$$\pi_8 = (124\ ,\ 35)$$

$$\pi_9 = (1\ ,\ 2\ ,\ 3\ ,\ 45)$$

$$\pi_{10} = (1\ 2\ 4\ 5\ ,\ 3).$$

L'ensemble CM des couples de partitions Mm pour l'automate dont la table de transitions est donnée en fig. 2.2 est :

$$CM = \{\pi_i \ , \ \pi_i'\} \qquad i = 1, 2, \ldots, 10.$$

La partition $m(\pi)$ traduit le maximum d'informations que l'on peut avoir sur l'automate, connaissant seulement π.

De même, la partition $M(\pi')$ représente le minimum d'informations nécessaire au calcul de π'.

3) Calcul de l'ensemble CS des couples de partitions à propriété de substitution :

Une partition π possède la propriété de substitution si (π,π) est un couple de partitions .

On a alors :

$$\pi = M(\pi) = m(\pi).$$

Soit π^s_{ij} la plus petite partition possédant la propriété de substitution et dans laquelle i et j sont dans un même bloc ; utilisant cette hypothèse et l'application f, on détermine tous les blocs de π^s_{ij}. On répète le même processus pour tout couple (i,j) non ordonné d'états de l'automate.

Pour un automate à p états, on aura au maximum $p\,\frac{(p-1)}{2}$ essais.

Comme dans le cas des couples de partitions Mm, la génération des partitions avec propriété de substitution se fait

en deux étapes :

a) Déterminer toutes les partitions π^s_{ij} *par la méthode précédente.*

b) Trouver toutes les sommes distinctes $\Sigma\ \pi^s_{ij}$.

L'ensemble CS des partitions à propriété de substitution (P S) est formé des partitions obtenues aux deux étapes.

Exemple de calcul de CS : Soit un automate A dont la table de transitions est donnée en figure 2.3 :

	x_1	x_2	x_3
1	6	3	2
2	5	4	1
3	2	5	4
4	1	6	3
5	4	1	6
6	3	2	5

Fig. 2.3

Si π^s_{12} identifie les états 1 et 2, alors :

$$f(1,x_1) \equiv f(2,x_1) \Rightarrow 6 \equiv 5$$
$$f(1,x_2) \equiv f(2,x_2) \Rightarrow 3 \equiv 4$$
$$f(1,x_3) \equiv f(2,x_3) \Rightarrow 1 \equiv 2$$

donc $\pi^s_{12} \geqslant (12\ ,\ 34\ ,\ 56)$

$$f(3,x_1) \equiv f(4,x_1) \Rightarrow 1 \equiv 2$$
$$f(3,x_2) \equiv f(4,x_2) \Rightarrow 5 \equiv 6$$

$$f(3,x_3) \equiv f(4,x_3) \Rightarrow 3 \equiv 4$$

$$f(5,x_1) \equiv f(6,x_1) \Rightarrow 3 \equiv 4$$

$$f(5,x_2) \equiv f(6,x_2) \Rightarrow 1 \equiv 2$$

$$f(5,x_3) \equiv f(6,x_3) \Rightarrow 5 \equiv 6$$

On a donc :

$$\pi^s_{12} = (12\ ,\ 34\ ,\ 56)\ .$$

De même :

$$\pi^s_{13} = (135\ ,\ 246)$$

$$\pi^s_{14} = I$$

$$\pi^s_{15} = \pi^s_{13}$$

$$\pi^s_{16} = \pi^s_{14} = \pi^s_{23} = \pi^s_{25} = \pi^s_{36} = \pi^s_{45} = I$$

$$\pi^s_{24} = \pi^s_{13} = \pi^s_{26} = \pi^s_{35} = \pi^s_{46}$$

$$\pi^s_{34} = \pi^s_{56} = \pi^s_{12}\ .$$

On obtient dans cette première étape deux partitions PS : π^s_{12} et π^s_{13} .

Par la deuxième étape, on obtient $\pi^s_{12} + \pi^s_{13} = (123456) = I$. Pour l'automate considéré, on a seulement deux partitions PS non triviales : π^s_{12} et π^s_{13} .

$$CS = \{\ \pi^s_{12}\ ,\ \pi^s_{13}\}\ .$$

II.3.5. Les recouvrements [6] [7] .

Définition :

Soit Y l'ensemble des états d'un automate et soit $\psi = \{C_i\}$ une collection de sous-ensembles de Y vérifiant

a) $Y = \Sigma\, C_i$

b) $C_i \subseteq C_j \iff i = j$.

Un ensemble ψ, satisfaisant les deux relations précédentes, est un recouvrement de Y.

Les éléments C_i sont les blocs du recouvrement. On peut noter que toute partition est un cas particulier de recouvrement.

Transformée d'une partition par la réduction :

Soit $Y = \{y_1, y_2, y_3, y_4, \ldots, y_8\}$ l'ensemble des états d'un automate non réduit, dont les états équivalents sont :

$$y_1 \sim y_5 \quad ; \quad y_2 \sim y_6 .$$

Soit $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ l'ensemble des états de l'automate réduit tels que : p_1 correspond à y_1 et y_5, p_2 à y_2 et y_6, p_3 à y_3, p_4 à y_4, p_5 à y_7 et p_6 à y_8.

D'autre part, on définit sur Y la partition π à propriété de substitution :

$$\pi = (y_1 y_2 y_3 y_4 \ , \ y_5 y_6 y_7 y_8).$$

A cette partition sur Y correspond une décomposition de l'ensemble P des états de l'automate réduit :

$$R(\pi) = \{p_1 p_2 p_3 p_4 \ , \ p_1 p_2 p_5 p_6\} = (C_1 \ , \ C_2).$$

On voit que la réduction des états d'un automate peut conduire à une perte d'information concernant la structure d'un système séquentiel.

Propriété : L'ensemble des recouvrements d'un ensemble constitue un treillis.

II-4 APPLICATION DES PARTITIONS ET DES RECOUVREMENTS A LA DECOMPOSITION DES AUTOMATES.

II.4.1 Cas des partitions.

II.4.1.1. Décomposition série d'un automate :

Théorème de Hartmanis :

Un automate A est décomposable en deux automates A_1 et A_2 connectés en série ($A_1 \Theta A_2$) si et seulement si il existe une partition π à propriété de substitution sur l'ensemble Y des états de A.

Soit π la partition P S dont le plus grand bloc à k éléments et soit τ la partition à k blocs, telle que :

$$\pi . \tau = 0.$$

En effet, le produit des deux partitions doit permettre de définir sans ambiguïté tous les états.

Les automates A_1 et A_2 sont définis par :

$$A_1 = (X, B\pi , X.B\pi , f_1, g_o)$$
$$A_2 = (X.B\pi , B\tau , Z, f_2, g_2)$$

où

$B\pi = \{b\pi\}$ est l'ensemble des blocs de la partition π,
f_1 est la restriction de f à $B\pi$,

g_o est l'application identique,

$B\tau = \{b\tau\}$ est l'ensemble des blocs de la partition τ;

avec $f_2(b\tau\ ,(b\pi\ ,x)) = f(b\tau \cap b\pi,x)$

$g_2(b\tau\ ,(b\pi\ ,x)) = g(b\tau \cap b\pi,x)$.

Ainsi l'automate A_1 calcule le bloc de π qui contient l'état présent de l'automate à décomposer A. Comme les états de A_2 sont les blocs de τ , les entrées à A_2 sont un bloc de π et un élément de X, l'état présent de A et son entrée sont connus ; A_2 peut calculer le bloc de τ qui contient l'état suivant de A. Pour stocker ce résultat, on a besoin de k mémoires élémentaires.

Exemple : Soit l'automate A donné par la figure 2.4 :

$A = (X,Y,Z,f,g)$

$X = \{0,1\}$

$Y = \{1,2,3,4,5\}$

	0	1
1	5/1	3/0
2	3/0	4/0
3	1/0	5/1
4	2/0	3/0
5	1/0	4/0

Fig. 2.4

On voit que $\pi = (12\ ,\ 345) = (A',B)$ est une partition P S. Donc A peut être décomposé en deux automates A_1 et A_2 ayant chacun deux et trois états respectivement.

La construction de A_1 est donnée sur la figure 2.5

	0	1
A'	B/0	B/1
B	A'/0	B 1

Fig. 2.5

La détermination de A_2 nécessite le choix de τ telle que $\pi\ .\ \tau = 0$.

On prend $\tau = (13\ ,\ 24\ ,\ 5) = (a\ ,\ b\ ,\ c)$.

Pour déterminer A, on réécrit la table de fluence de la figure 2.4 dans laquelle on remplace les états de A par les blocs correspondants de τ et en réarrangeant les informations on obtient la table de fluence de A_2 de la figure 2.6.

	0	1
a	c/1	a/0
b	a/0	b/0
.....	.....	.....
a	a/0	c/1
b	b/0	a/0
c	a/0	b/0

a)

	A'0	A'1	B0	B1
a	c/1	a/0	a/0	c/1
b	a/0	b/0	b/0	a/0
c	-/-	-/-	a/0	b/0

b)

0	1	0	1
B	B	α	β
A'	B	β	α

c)

	α	β	α	β
a	c	a	1	0
b	a	b	0	0
c	b	a	0	0

d)

Fig. 2.6

En choisissant convenablement les états indifférents, on obtient deux colonnes identiques dans la figure 2.6.b. Comme on a seulement besoin de deux entrées pous identifier les deux colonnes différentes, on peut simplifier la sortie de A_1 qui prend les valeurs α et β .

Les sous-automates simplifiés A_1 et A_2 sont donnés sur la figure 2.6.c et d.

II.4.1.2. Généralisation : décomposition sans boucle :

Le schéma de principe de la décomposition d'un automate A en n sous-automates A_i est représenté par la figure 2.7. L'information se déplace dans un seul sens de l'entrée vers la sortie.

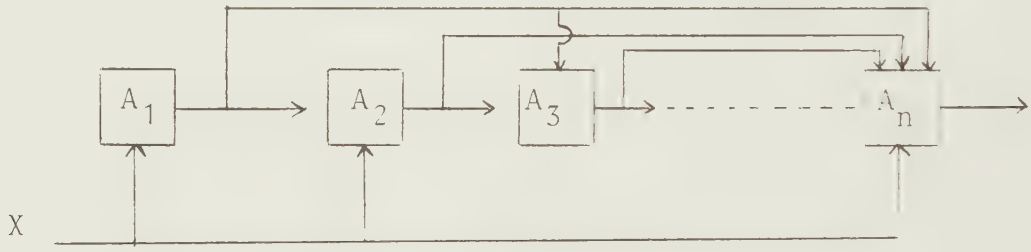


Fig. 2.7
Décomposition d'un automate en n sous-automates sans contre-réaction.

L'examen de la figure 2.7 montre que l'état suivant d'un sous-automate A_i dépend seulement des automates A_j où $j \leq i$.

Un automate A admet une décomposition sans boucle en n sous-automates A_k s'il n'existe aucun couple A_i et A_j de sous-automates de A tels que les sous-ensembles fermés des prédécesseurs de A_i et A_j étant $C(A_i)$ et $C(A_j)$, on ait :

$$A_i \subset C(A_j) \qquad \text{et} \qquad A_j \subset C(A_i).$$

Théorème de Hartmanis :

Soit $T = \{\pi_1, \pi_2, \ldots, \pi_n\}$ un ensemble de partitions à propriété de subsitution définies sur l'ensemble des états Y d'un automate A telles que :

$$\prod_{i=1}^{n} \pi_i = 0.$$

Il existe alors n sous-automates A_i constituant une décomposition de l'automate A sans boucle.

Pour effectuer la décomposition de l'automate A, on choisit un ensemble $T = \{ \pi_1, \pi_2, \ldots, \pi_n \}$ défini précédemment.

Si le seul ensemble T qui existe est T = (I,O), on ne peut décomposer A sous la forme annoncée.

La décomposition de l'automate A se fait de la façon suivante :

a) on choisit l'ensemble T_1 de toutes les partitions π_i de T qui ne sont pas plus petites que toute autre partition dans T. Les sous-automates A_i associés aux partitions $\pi_i \in T_1$ opèrent en parallèle(§ II.4.1.3.).

Si $T_1 = T$ on a décomposé A en sous-automates fonctionnant en parallèle. Sinon on considère l'ensemble $T-T_1$ des autres partitions.

b) On choisit l'ensemble T_2 des partitions π_i de $T-T_1$ qui ne sont pas plus petites que d'autres.

Pour tout $\pi_j \in T_2$, on considère les partitions $\pi_k \in T_1$ telles que :

$$\pi_j \leq \pi_k .$$

Les sous-automates A_k relatifs aux π_k, procurent des entrées au sous-automate A_j relatif à π_j.

La partition π_j^* :

$$\pi_j^* = \Pi\ \pi_k \quad \text{tels que} \quad \pi_j < \pi_k \in T_1$$

identifie les informations provenant des A_k et servant comme entrée à A_j.

Soit τ_j la partition identifiant les états de A_j telle que :

$$\pi_j = \pi_j^* \cdot \tau_j \ .$$

La partition τ_j doit contenir l blocs, l étant le plus grand nombre de blocs de π_j contenus dans un seul bloc de π_j^* afin que τ_j contienne un nombre suffisant d'états différents pour que l'on ait $\pi_j^* \tau_j = \pi_j$.

Le choix de τ_j n'est pas unique.

Pour une partition τ_j choisie, on considère le couple (π_j , τ_j) et la fonction de transition f de l'automate à décomposer, pour déterminer la fonction f_j correspondant à A_j.

Si τ_j ne transforme pas tout bloc de π_j^* en un même nombre de blocs, A_j n'est pas complètement défini.

c) On continue le processus précédent jusqu'à épuisement des partitions. Comme $\Pi\ \pi_i = 0$, l'automate composé des A_i interconnectés de la façon précédente est équivalent à l'automate A.

Exemple : Soit l'automate A défini par la table de transition de la figure 2.8.

	x_1	x_2	x_3	x_4
1	2/0	3/1	4/1	1/0
2	1/1	3/0	4/1	1/0
3	2/1	1/1	4/0	1/0
4	2/0	3/0	1/1	1/1

Fig. 2.8
Table de transition de A

Les partitions à propriété de substitution sont :

$0 = (1, 2, 3, 4)$
$\pi_1 = (123, 4)$
$\pi_2 = (134, 2)$
$\pi_3 = (124, 3)$
$\pi_4 = (12, 3, 4)$
$\pi_5 = (13, 2, 4)$
$\pi_6 = (14, 2, 3)$

Il y a plusieurs ensembles T possibles que l'on peut former à partir des partitions $0, \pi_1, \pi_2, \ldots, \pi_6$.

Le choix de $T = \{\pi_1, \pi_2, \pi_4\}$ est acceptable.

En effet : $\pi_1 \cdot \pi_2 \cdot \pi_4 = 0$.

L'examen des partitions π_1 et π_2 montre qu'elles ne sont pas contenues dans d'autres partitions, donc :

$$T_1 = \{ \pi_1 , \pi_2 \}.$$

On peut définir deux sous-automates A_1 et A_2 respectivement par π_1 et π_2 et fonctionnant en parallèle.

Codage des états de A_1 et A_2 et fonctions de transitions :

Chacun des deux blocs de π_1 et π_2 est codé respectivement par y_{11}, y_{12} et y_{21}, y_{22} et les fonctions de transitions correspondantes sont (Fig. 2.9) :

Y_1	x_1	x_2	x_3	x_4
y_{11}	y_{11}	y_{11}	y_{12}	y_{11}
y_{12}	y_{11}	y_{11}	y_{11}	y_{11}

Y_2	x_1	x_2	x_3	x_4
y_{21}	y_{22}	y_{21}	y_{21}	y_{21}
y_{22}	y_{21}	y_{21}	y_{21}	y_{21}

Fig. 2.9 - Fonctions de transitions de A_1 et A_2

On considère l'ensemble $T - T_1 = \{\pi_4\}$. Cet ensemble étant formé d'une seule partition, on prend $T_2 = \{\pi_4\}$. La comparaison des partitions de T_1 et celle de T_2 montre que :

$\pi_4 \leqslant \pi_1$ et π_2 et π_4 ne sont pas comparables.

Le sous-automate A_1 calcule des entrées de l'automate A_3 relatif à la partition π_4.

Pour déterminer les états de A_3, on doit trouver une partition τ_3 telle que $\pi_1 . \tau_3 = \pi_4$ ($\pi_3^* = \pi_1$).

En choisissant τ_3, on sait que le nombre maximum de ses blocs est deux puisque on a deux blocs au maximum de π_4 qui sont inclus dans un seul bloc de π_1. Une telle partition est $\tau_3 = (12 , 34)$.

Le couple de partition $(\pi_3^* \tau_3, \tau_3)$ sert à définir la fonction état suivant de A_3. En codant les deux blocs de τ_3 par $Y_3 = (y_{31}, y_{32})$, on obtient la table de transition de A_3 (figure 2.10) :

Y_3 \ $Y_1 . X$	$y_{11}x_1$	$y_{11}x_2$	$y_{11}x_3$	$y_{11}x_4$	$y_{22}x_1$	$y_{22}x_2$	$y_{22}x_3$	$y_{22}x_4$
y_{31}	y_{31}	y_{32}	y_{32}	y_{31}	-	-	-	-
y_{32}	y_{31}	y_{31}	y_{32}	y_{31}	y_{31}	y_{32}	y_{31}	y_{31}

Fig. 2.10

Table de transitions de A_3.

La présence des états indifférents tient au fait que τ_3 ne transforme pas tout bloc de π_1 en un même nombre de plus petits blocs.

La correspondance entre l'automate A et les sous-automates A_i composants est donnée sur la figure 2.11 et est obtenue directement à partir des partitions de $T = \{\pi_1, \pi_2, \pi_4\}$. L'état 1 de A est équivalent à y_{11}, y_{21}, y_{31} car 1 est contenu dans les blocs 123 de π_1, 134 de π_2 et 12 de τ_3.

états de A	entrées / états de A_c	sortie			
		x_1	x_2	x_3	x_4
1	$y_{11}y_{21}y_{31}$	0	1	1	0
2	$y_{11}y_{22}y_{31}$	1	0	1	0
3	$y_{11}y_{21}y_{32}$	1	1	0	0
4	$y_{12}y_{21}y_{32}$	0	0	1	1

Fig. 2.11

Correspondance entre l'automate composé A_c et A.

On peut remarquer dans ce tableau que seuls 4 des 8 états de l'automate composé A_c sont utiles. Cela tient au fait que A est équivalent à un sous-automate de A_c. Les autres états de A_c sont sans importance tant que l'état initial de A_c est l'un de ceux des figures 2.9 ou 2.10.

II.4.1.3. Décomposition parallèle d'un automate :

Définition :

Un automate $A = (X,Y,Z,f,g)$ admet une décomposition parallèle en deux automates A_1 et A_2 $(A_1 \| A_2)$ si :

$$A_1 = (X_1,Y_1,Z_1,f_1,g_1)$$
$$A_2 = (X_2,Y_2,Z_2,f_2,g_2)$$
$$X = X_1 \cdot X_2$$
$$Y = Y_1 \cdot Y_2$$
$$Z = Z_1 \cdot Z_2$$

$$f\,[(y_1,y_2),(x_1,x_2)] = (f_1(y_1,x_1),f_2(y_2,x_2))$$
$$g\,[(y_1,y_2),(x_1,x_2)] = (g_1(y_1,x_1),g_2(y_2,x_2))$$

Théorème de Hartmanis : Un automate A admet une décomposition parallèle en automates A_1 et A_2 ($A_1 || A_2$) si et seulement si il existe deux partitions à propriété de substitution π_1 et π_2 telles que :

$$\pi_1 \cdot \pi_2 = 0.$$

Exemple : Soit A = (X,Y,Z,f,g) dont la table de transitions est donnée figure 2.12.

	0	1
1	4	3
2	6	3
3	5	2
4	2	5
5	1	4
6	3	4

Fig. 2.12
Table de transitions de l'automate A.

Pour cet automate, les partitions π_1 = (123,456) et π_2 = (16,25,34) ont la propriété de substitution et $\pi_1 \cdot \pi_2 = 0$. On peut donc décomposer A en deux automates A_1 et A_2 ayant respectivement deux et trois états. Les tables de transition de ces deux automates sont représentées en figure 2.13.

	0	1
y_{11}	y_{12}	y_{11}
y_{12}	y_{11}	y_{12}

	0	1
y_{21}	y_{23}	y_{23}
y_{22}	y_{21}	y_{23}
y_{23}	y_{22}	y_{22}

Fig. 2.13
Tables des transitions des automates A_1 et A_2.

[illegible]

Définition :

La décomposition avec boucle fermée d'un automate A en sous-automates A_i est représentée schématiquement sur la figure 2.14 : l'information émanant d'un automate est injectée à l'entrée d'un automate précédent.

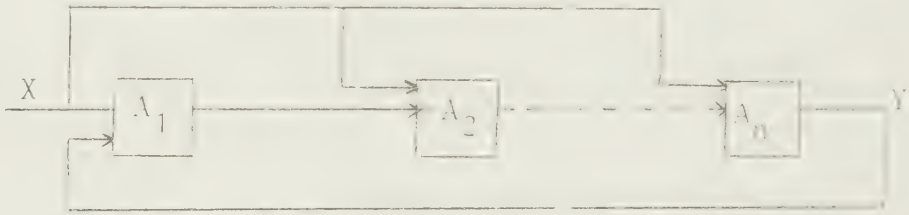


Fig. 2.14
Décomposition d'un automate avec boucle fermée.

Méthode :

Soient CP l'ensemble de tous les couples de partitions d'un automate A et $F = \{(\pi_i, \tau_i)\}$ un sous-ensemble de CP. La partition τ_i sert à définir les états de l'élément composant A_i et π_i contient les informations nécessaires à la définition de la fonction état suivant pour le sous-automate A_i.

Soit $F = \{(\pi_i, \tau_i)\}$ un sous-ensemble de CP à k couples de partitions tel que $\prod_{i=1}^{k} \tau_i = 0$.

Pour toute partition τ_i, $M(\tau_i)$ représente le minimum d'informations amont à partir desquelles on peut calculer les blocs de τ_i dans lesquels se trouvent les états de A_i. On prendra $\pi_i = M(\tau_i)$.

π_i détermine les entrées de A_i.

On doit trouver toutes les partitions τ_j telles que

$$\tau_j \leq \pi_i$$
$$\tau_j \cdot \tau_m \leq \pi_i \qquad 1 \leq j, m, l \leq k$$
$$\tau_j \cdot \tau_m \cdot \tau_l \leq \pi_i$$

Comme $\prod_{i=1}^{k} \tau_i = 0$, le processus précédent est fini. En particulier, si $\tau_i \leqslant \pi_i$, les seules entrées de A_i sont les commandes extérieures.

Une fois l'étape précédente terminée, on détermine la fonction état suivant pour chaque composant A_i.

Exemple :

Soit A = (X,Y,Z,f,g) un automate dont la table de transitions est donnée figure 2.15.

	x_1	x_2	x_3	x_4
1	2	1	5	1
2	2	1	5	2
3	3	5	2	4
4	3	5	4	1
5	3	5	4	2

Fig. 2.15

Table de transitions de l'automate A.

Les couples de partitions Mm sont :

$$(\pi_1,\pi_1') = \{(12,3,4,5) \ , \ (124,3,5)\}$$

$$(\pi_2,\pi_2') = \{(14,3,25) \ , \ (145,23)\}$$

$$(\pi_3,\pi_3') = \{(12,45,3) \ , \ (12,3,4,5)\}$$

$$(\pi_4,\pi_4') = \{(35,1,2,4) \ , \ (24,1,3,5)\} \ .$$

Il y a plusieurs façons de choisir les couples de partitions nécessaires à la description du comportement de l'automate. Toutefois, le choix doit être guidé par le souci d'avoir un nombre minimum d'état d'une part, et d'autre part la dépendance entre les composants doit être réduite. Un des choix possibles pour les partitions τ est :

$$\tau_1 = (124 \ , \ 35)$$

$$\tau_2 = (145 \ , \ 23)$$

$$\tau_3 = (12 \ , \ 345).$$

Ce choix est acceptable car $\prod_{i=1}^{3} \tau_i = 0$.

Pour déterminer les $M(\tau_i)$, on utilise les partitions Mm :

$$\tau_1 \geqslant \pi'_1$$

$$\tau_2 \geqslant \pi'_2$$

$$\tau_3 \geqslant \pi'_3$$

Les partitions associées sont :

$$(\pi_1,\tau_1) \quad ; \quad (\pi_2,\tau_2) \quad ; \quad (\pi_3,\tau_3) \ .$$

Comme $\tau_3 \leqslant \pi_1$, A_3 produit une entrée à A_1. En testant π_2 et π_3, on voit qu'aucune partition τ_i n'est plus petite que ces deux partitions.

On essaye alors les produits $\tau_j.\tau_1$ pour déterminer les entrées à A_1 et A_2. On voit que $\tau_1.\tau_2 \leqslant \pi_2$ et $\tau_2.\tau_3 \leqslant \pi_3$, donc les fonctions états suivant des deux sous-automates dépendent respectivement de A_1 et A_2 pour A_2 et de A_2 et A_3 pour A_3.

En utilisant ces résultats, on obtient les fonctions états suivant des automates composants (figure 2.16).

Y_1 \ $X.Y_3$	x_1,y_{31}	x_2,y_{31}	x_3,y_{31}	x_4,y_{31}	x_1,y_{32}	x_2,y_{32}	x_3,y_{32}	x_4,y_{32}
y_{11}	y_{11}	y_{11}	y_{12}	y_{11}	y_{12}	y_{12}	y_{11}	y_{11}
y_{12}	y_{11}	y_{11}	y_{12}	y_{11}	y_{12}	y_{12}	y_{11}	y_{11}

Automate A_1

Y_2 \ $X.Y_1$	x_1,y_{11}	x_2,y_{11}	x_3,y_{11}	x_4,y_{11}	x_1,y_{12}	x_2,y_{12}	x_3,y_{12}	x_4,y_{12}
y_{21}	y_{22}	y_{21}	y_{21}	y_{21}	y_{22}	y_{21}	y_{21}	y_{22}
y_{22}	y_{22}	y_{21}	y_{21}	y_{22}	y_{22}	y_{21}	y_{22}	y_{21}

Automate A_2

Y_3 \ $X.Y_2$	x_1,y_{21}	x_2,y_{21}	x_3,y_{21}	x_4,y_{21}	x_1,y_{22}	x_2,y_{22}	x_3,y_{22}	x_4,y_{22}
y_{31}	y_{31}	y_{31}	y_{32}	y_{31}	y_{31}	y_{31}	y_{32}	y_{31}
y_{32}	y_{32}	y_{32}	y_{32}	y_{31}	y_{32}	y_{32}	y_{31}	y_{32}

Automate A_3

$$Y_1 = (y_{11},y_{12}) \rightarrow \tau_1 = (124,35)$$
$$Y_2 = (y_{21},y_{22}) \rightarrow \tau_2 = (145,23)$$
$$Y_3 = (y_{31},y_{32}) \rightarrow \tau_3 = (12,\ 345)$$

Fig. 2.16

Table de transitions des sous-automates A_i.

La représentation schématique est représentée figure 2.17.

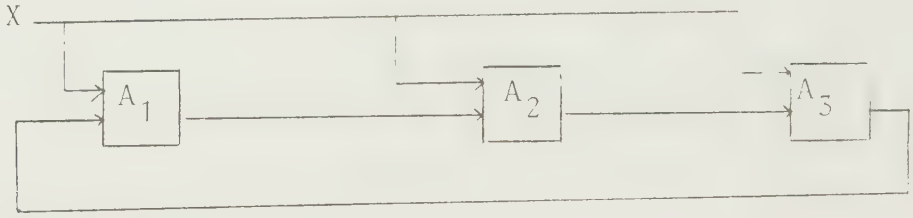


Fig. 2.17

Décomposition avec boucle fermée de A.

La décomposition n'est pas unique du fait que les partitions τ_i ne le sont pas.

II.4.2. Cas des recouvrements.

La méthode de décomposition des automates basée sur les recouvrements consiste à dédoubler des états d'un automate pour obtenir un automate équivalent et les recouvrements sont transformés en partitions. On obtient, soit des partitions à propriété de substitution, soit des couples de partitions suivant les recouvrements utilisés. Cette méthode est connue sous le nom d'"éclatement d'états".

On se ramène alors au cas précédent.

Exemple : Soit l'automate A défini par la table de transitions Figure 2.18.a :

états \ entrée	E_1	E_2
1	2	3
2	1	3
3	4	1
4	2	4

a)

états \ entrée	E_1	E_2
1	2	3
2	1	3
3	4	1
4	2	4'
4'	2	4

b)

Fig. 2.18

L'un des recouvrements de A ayant la propriété de substitution est :

$$\psi = (124,\ 34)$$

L'état 4 apparaît dans les deux blocs du recouvrement ψ ; on adjoint un état supplémentaire 4' aux états de A. On remplace l'état 4 du bloc 34 par le nouvel état 4' et l'on obtient une partition π à propriété de substitution :

$$\pi = (124,\ 34').$$

La table des transitions du nouvel automate A' équivalent à A est donnée fig. 2.18.b.

II.5 CONCLUSION.

On a donné dans ce chapitre les méthodes de décomposition d'un automate. Ces méthodes basées sur les partitions de l'ensemble Y des états d'un automate sont dues principalement à Hartmanis et Stearn [1] et permettent une décomposition structurelle. Les sous-automates ayant chacun ses propres états internes sont convenablement interconnectés et permettent, moyennant un choix convenable de l'état initial, de simuler l'automate décomposé.

Toutefois, les automates réels ne sont pas en général décomposables par ces méthodes et on peut noter également que, dans le cas d'une décomposition, les sous-automates A_i sont de taille variable. Pour répondre à cet inconvénient, nous introduisons dans le chapitre suivant un module universel, le module tétraèdrique, qui permet la décomposition des automates. Par ce procédé, nous établissons une décomposition technologique où chaque état interne de l'automate à décomposer est conservé.

CHAPITRE III

MODULE TETRAEDRIQUE

III-1 GRAPHE ATTACHE A UN AUTOMATE.

III.1.1. Définition d'un graphe.

Un graphe G est un couple (Y,Γ) où Y est un ensemble fini d'éléments, les sommets du graphe, et Γ la relation binaire traduisant le lien entre deux éléments y_i et y_j de Y.

On note :

$$G = (Y,\Gamma)$$

$\forall\, y_i$, $\forall\, y_j \in Y$: si $y_i \Gamma\, y_j$ est vraie, un arc orienté de y_i à y_j joint les deux sommets de Y (Fig. 3.1).

Fig. 3.1

L'arc orienté de y_i à y_j traduit le fait que $y_i \Gamma\, y_j$ est vraie.

III.1.2. Graphe non orienté.

Considérons un graphe $G = (Y,\Gamma)$:

La relation Γ est symétrique si lorsque $y_i \Gamma y_j$ existe, $y_j \Gamma y_i$ existe aussi.

La représentation graphique de cette relation est un ensemble de deux arcs ayant pour origine y_i et y_j. On représente aussi ces deux arcs par une arête joignant les deux sommets (fig. 3.2)

Fig. 3.2.

Arête joignant deux sommets y_i et y_j et traduisant la vérité de $y_i \Gamma y_j$ et $y_j \Gamma y_i$.

III.1.3. Sous-graphe d'un graphe.

Soit $G = (Y,\Gamma)$ un graphe : on appelle sous-graphe SG de G le couple :

$$SG = (Y',\Gamma) \quad \text{où} \quad Y' \subset Y.$$

III-2 AUTOMATE DE MEDVEDEV OU SEMI-AUTOMATE.

Soit un automate $A = (X,Y,Z,f,g)$; les équations de son fonctionnement sont (cf. Ch.I) :

$$Y_{n+1} = F(Y_n,X_n) \qquad (1)$$

$$Z_n = G(Y_n,X_n) \qquad (2)$$

On voit que seule la relation de récurrence du premier ordre (1) traduit le fonctionnement séquentiel du système et tient compte des états internes Y ou mémoires.

Définition : On appelle semi-automate le triplet :

$$MA = (X,Y,f)$$

où X,Y et f ont les mêmes significations que dans le quintuplet de Hartmanis. Ce triplet traduit simplement la relation de récurrence (1) ; on l'appelle aussi la machine des états.

III-3 GRAPHE ATTACHE A UN AUTOMATE.

Un graphe attaché à un automate est le couple GA défini par :

$$GA = (Y,\Gamma)$$

Y est l'ensemble des états de l'automate A

Γ est la relation binaire entre les éléments $y_i \in Y$, induite par l'application $f:Y.X \rightarrow Y$.

y_i et y_j étant deux éléments de Y :

$y_i \Gamma y_j$ est vraie si et seulement si il existe une entrée $x_k \in X$ telle que :

$$f(y_i,x_k) = y_j \qquad (y_j \text{ peut être égal à } y_i).$$

Dans la suite, pour étudier un automate, on traduira son fonctionnement par son graphe GA orienté. Ainsi :

a) l'étude d'un automate est réduite aux relations existant entre les éléments d'un seul ensemble Y ;

b) l'examen du graphe donne une idée de la complexité et de la structure de l'automate, la représentation graphique étant plus parlante qu'un tableau d'états surtout s'il contient beaucoup de vides, ce qui est le cas des systèmes séquentiels industriels.

III-4 RECHERCHE D'UN SOUS-GRAPHE MODULE.

III.4.1. Problème à résoudre.

Soit GA le graphe d'un automate ; est-il possible de trouver un sous-graphe permettant de recouvrir tout graphe GA et donc de décomposer l'automate A ?

Ce sous-graphe, suivant les transitions validées, doit s'identifier à tout graphe donné de même dimension.

Le sous-graphe module, s'il existe, doit répondre à trois critères :

1) être universel,
2) être peu encombrant,
3) être simplement programmable.

La construction d'un automate se fait par l'assemblage de modules identiques interconnectés entre eux par des liaisons câblées.

III.4.2. Dimension du sous-graphe module.

La dimension du sous-graphe module, en tenant compte des critères précédents, est prise égale à quatre.

La justification de ce choix réside dans le fait qu'un module à cinq états internes ou plus est assez gros et donc trop complexe, et par contre, un module à trois états est trop petit. Dans le premier cas, la programmation nécessite un grand nombre de commandes, et pour une synthèse modulaire cela augmente les liaisons entre modules, ce qui n'est pas souhaitable pour deux raisons :

a) les liaisons intermodulaires sont à câbler pour chaque système construit à partir des modules ;

b) dans le cas d'une panne, les modules défectueux sont facilement détectables si les connexions entre modules sont réduites.

Une dimension intermédiaire, quatre, paraît être raisonnable.

La programmation du graphe module permet de construire toute configuration de graphe à quatre sommets et donc de traduire les transitions entre quatre états quelconques d'un automate.

III.4.3. Sous-graphe module.

Le sous-graphe module est alors représenté en figure 3.3. La relation binaire Γ entre les quatre sommets est symétrique. L'information peut se propager de n'importe quel sommet vers un autre.

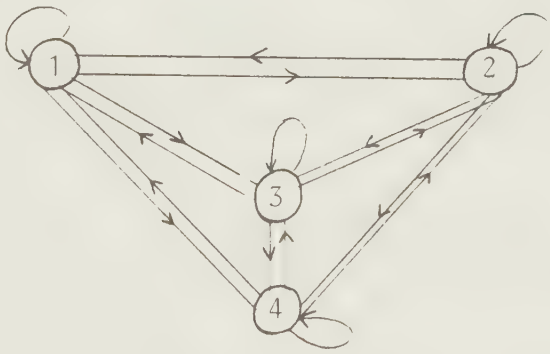


Fig. 3.3
Sous-graphe module.

La relation binaire Γ est symétrique :

$$\forall i \text{ et } j \in \{1,2,3,4\} \quad : \left.\begin{array}{c} i\Gamma j \\ \text{et} \\ j\Gamma i \end{array}\right\} \text{ sont vraies.}$$

Les éléments i et j peuvent être distincts ou non.

Inconvénient :

Le graphe de la figure 3.3 ne peut communiquer avec le milieu extérieur.

Sous-graphe module complet :

Afin d'éviter l'isolement du module et de rendre opérationnel le sous-graphe module, on introduit deux sommets fictifs S et P qui rendent compte respectivement des informations reçues et émanant de chaque état de l'automate correspondant au sous-graphe module à quatre états internes 1,2,3 et 4. On obtient le sous-graphe module complet figure 3.4. Chacun des deux sommets fictifs S et P est, en réalité, un sommet d'un ou des modules voisins du module considéré.

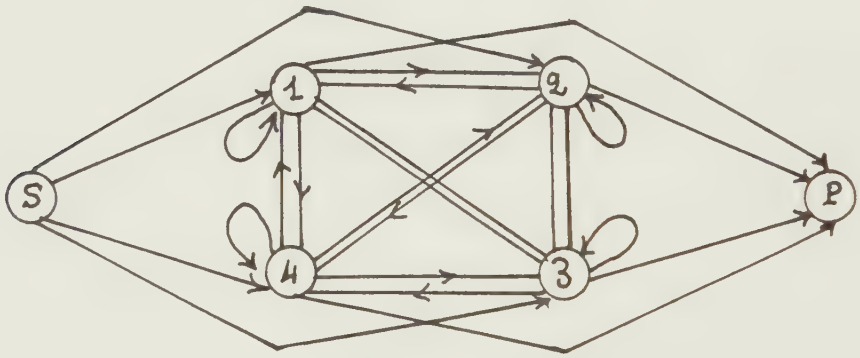


Fig. 3.4
Sous-graphe module complet

Programmation du module :

Elle consiste, en partant du sous-graphe module complet, à construire tout graphe à quatre sommets. Cette construction s'effectue en choisissant convenablement les commandes, éléments de X, alphabet d'entrée, à appliquer à chaque sommet pour provoquer les transitions voulues ; les sommets fictifs S et P servent à l'échange de l'information entre les modules.

III-5 MODULE TETRAEDRIQUE.

III.5.1. Définition.

C'est l'automate à quatre états internes dont le graphe attaché est représenté sur la figure 3.4.

III.5.2. Codage des états du module.

a) Codage binaire biunivoque des états internes du module :

Pour coder binairement N états, si aucune contrainte n'est imposée, il suffit de n variables internes ou mémoires telles que :

$$n = [\text{Log}_2 N] \ .$$

Le deuxième terme de l'égalité désignant un nombre entier égal à $\log_2 N$, si ce nombre est entier, sinon la partie entière de $\log_2 N+1$.

Ainsi, dans le cas de N = 4, le nombre de mémoires est n = 2. Pour le module, ce codage est non acceptable ; en effet :

codons les états 1,2,3,4 par la suite binaire 00,01,11,10, figure 3.5.

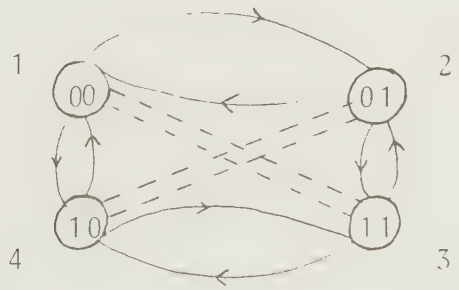


Fig. 3.5
Essai de codage des quatre états.

Les transitions 1-3 et 2-4 ne sont pas adjacentes et cela peut conduire, compte-tenu du support physique du codage, à des courses critiques. La contrainte d'adjacence peut être surmontée en synchronisant les mémoires correspondant aux variables internes par un signal d'horloge .

b) Codage un parmi N :

C'est un codage canonique, un parmi quatre. A chaque état i est associée une variable interne y_i représentée par l'état d'une mémoire élémentaire. L'automate est dans l'état i si et seulement si $y_i = 1$.

III-6 CELLULE ELEMENTAIRE ASSOCIEE A UN ETAT i.

Soient :

$$P_i = \{p_1, p_2, \ldots, p_n\} \qquad P_i \subset Y$$

$$S_i = \{s_1, s_2, \ldots, s_r\} \qquad S_i \subset Y$$

les ensembles des prédécesseurs et des successeurs de l'état i d'un automate A. La figure 3.6 représente une partie du graphe des transitions relatives à l'état i de A.

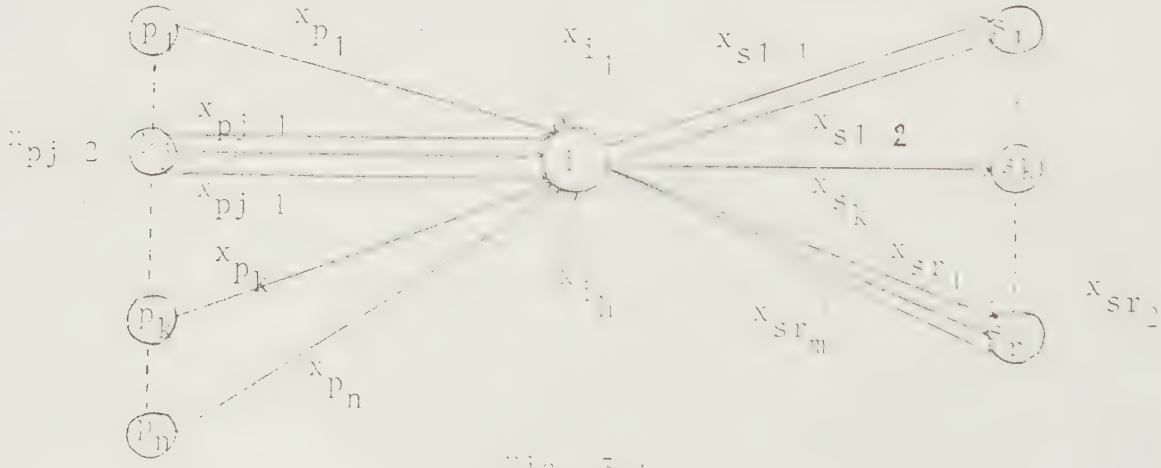


Fig. 3.6

Graphe de transitions d'un automate A relatif à l'état i

On voit sur ce graphe de transitions que la commande x_{p_1} fait passer le système de l'état p_1 à l'état i ; de même les entrées différentes $x_{p_{j\,1}}$, $x_{p_{j\,2}}$, $x_{p_{j\,5}}$ provoquent la même transition de l'état p_j à l'état i ; les commandes $x_{i\,1}$, $x_{i\,2}$ sont inactives et ne provoquent aucune transition ; les commandes $x_{sr\,1}$, $x_{sr\,2}$, $x_{sr\,m}$ permettent d'effectuer la transition de i à s_r.

On peut résumer ce qui précède par les deux relations suivantes :

1) Activation de la cellule :

Pour que la cellule élémentaire soit activée (niveau haut), il faut que l'on soit d'abord dans l'un des états prédécesseurs, tel que p_k et que l'on applique une commande x_{p_k}.

La traduction de cette condition est donnée par l'équation (1) :

$$y_i(n+1) = \sum_{k=1}^{N} \left[y_{pk}(n) \left(\sum_{j=1}^{l} x_{pkj}(n) \right) \right] + y_i(n) \left(\sum_{t=1}^{h} x_{it}(n) \right) \quad (1)$$

où le second terme de la somme correspond aux boucles quand elles existent.

2) Inhibition de la cellule élémentaire :

Lors de la transition de l'état i à tout état successeur s_k, la cellule représentant l'état i doit être remise au niveau bas. La condition (2) de remise à zéro s'exprime par :

$$\bar{y}_{i\,(n+1)} = y_{i\,(n)} \left[\sum_{k=1}^{r} x_{s_k}(n) + \sum_{t=1}^{m} x_{s_{kt}}(n) \right] \quad (2)$$

Les premiers termes des relations (1) et (2) correspondent aux commandes S et R respectivement de mise à un et à zéro d'une mémoire élémentaire représentant l'état i.

III-7 STRUCTURE DE LA CELLULE ELEMENTAIRE REPRESENTANT UN ETAT.

Dans sa forme primitive, la cellule est constituée par une mémoire M pouvant être mise à 1 ou 0 selon les commandes S et R.

A partir de cette mémoire M on peut aller vers les trois autres cellules voisines par les commandes x,y,z respectivement ou vers un autre module par la commande h . De même, on peut venir dans la mémoire à partir d'un autre module ou bien à partir des trois états voisins situés dans le même module par les commandes respectives E, V_1, V_2, V_3 (figure 3.7)

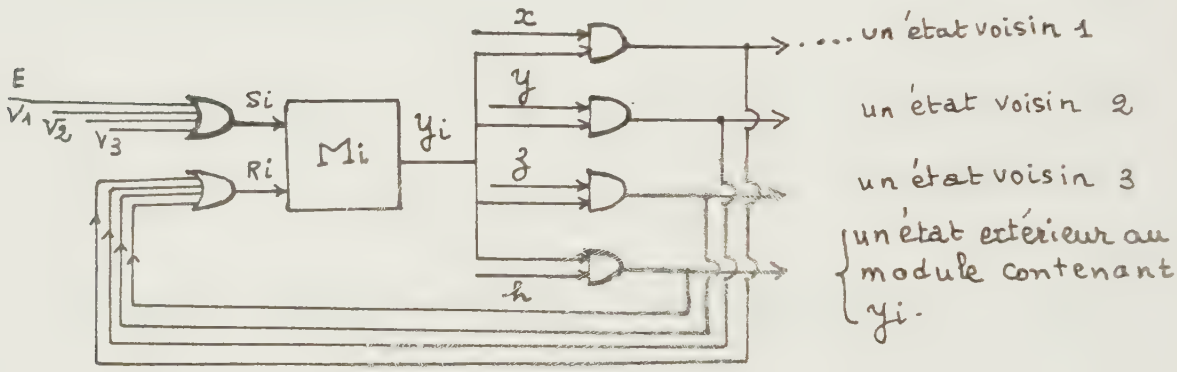


Fig. 3.7

Schéma de base d'une cellule élémentaire.

Le passage d'un état interne y_i à un autre état y_j (éventuellement i = j) se fait sous l'action d'une seule entrée ou variable primaire du type $x_{p\,kj}$ ou $x_{s\,kt}$, comme l'indique la relation (1). L'action de ces entrées est traduite par une des commandes intermédiaires :

$$x,y,z,h = \sum_{j=1}^{l} x_{p\,kj}$$

- S'il existe une boucle, c'est-à-dire si sous l'action d'un vecteur d'entrée x_{ik} l'état interne reste le même, on n'envoie pas de commande.

- A chaque transition vers un état voisin ou un état extérieur au module, est associée une des commandes x,y, z ou h.

La commande S_i qui force la mémoire M_i à l'état 1 résulte de l'action combinée d'un y_{p_k} (état prédécesseur) et d'une commande.

La commande R_i de remise à zéro de la mémoire M_i résultè de l'action combinée de y_i et de la même commande.

Les expressions (1) et (2) donnent alors :

pour la mise à 1 :

$$S_i = \sum_{k=1}^{n} \left[y_{p_k} \left(\sum_{j=1}^{l} x_{p_{kj}} + x_{p_k} \right) \right] \qquad (1)$$

pour la remise à zéro :

$$R_i = y_i \left[\sum_{k=1}^{r} \left(x_{s_k} + \sum_{t=1}^{m} x_{s_{kt}} \right) \right] \qquad (2)$$

III-8 STRUCTURE DU MODULE TETRAEDRIQUE.

Le module tétraédrique est l'ensemble de quatre cellules identiques analogues à celle que l'on vient de décrire et interconnectées conformément au graphe de la figure 3.4 (boucles exceptées).

III-9 COMMANDES DU MODULE TETRAEDRIQUE.

Nombre de commandes :

Un module tétraédrique est un automate à quatre états dont le codage est assuré par quatre cellules élémentaires.

Pour chaque cellule, on doit avoir :

- une entrée E pour la lecture de l'information (mise à 1 de la mémoire) ;
- quatre entrées primaires x,y,z,h pour l'écriture de l'information en permettant d'effectuer les transitions vers les trois autres états voisins dans un même module ou dans un état situé dans un autre module.

Pour les quatre cellules, le nombre de commandes est vingt : E_i, x_i, y_i, z_i, h_i avec $i = 1,2,3,4$.

Réduction du nombre de commandes :

Les variables secondaires d'entrée E_i ne peuvent être réduites du fait de la symétrie du module, sinon celui-ci se trouve isolé.

Un choix judicieux des interconnexions des cellules permet d'utiliser seulement quatre variables primaires d'entrée. Ce nombre est optimal. La figure 3.8 met en relief le rôle des quatre variables x,y,z et h.

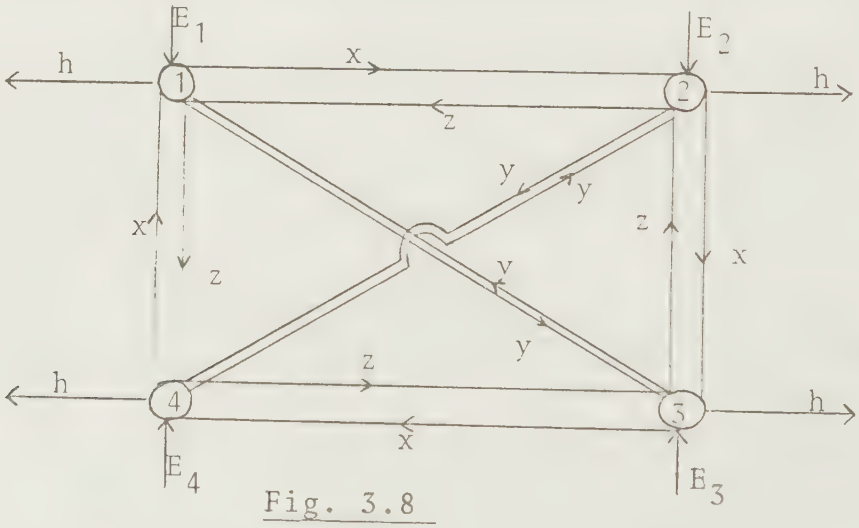


Fig. 3.8

Organisation des variables primaires et secondaires du module tétraédrique.

- La variable secondaire E_i $(i = 1,2,3,4)$ permet l'entrée de l'information dans le module $(y_i = 1)$.

- La variable primaire x permet une circulation de l'information dans le sens 1-2-3-4-1 entre les états d'un même module.

- La variable primaire z permet une circulation dans le sens inverse du précédent 1-4-3-2-1.

- La variable primaire y permet l'échange d'information entre les états 1 et 3 d'une part et 2 et 4 d'autre part.

Les équations logiques du paragrapheIII-7 pour un état i du module $(i = 1,2,3,4)$ sont :

<u>pour l'état 1</u> :

Activation de la mémoire M1 :

$$S_1 = E_1 + y_4x + y_3y + y_2z$$

Remise à zéro de M_1 :

$$R_1 = y_1(x + y + z + h)$$

<u>pour les autres états, on a</u> :

$$\begin{cases} S_2 = E_2 + y_1x + y_4y + y_3z \\ R_2 = y_2(x + y + z + h) \end{cases}$$

$$\begin{cases} S_3 = E_3 + y_2x + y_1y + y_4z \\ R_3 = y_3(x + y + z + h) \end{cases}$$

$$\begin{cases} S_4 = E_4 + y_3x + y_2y + y_1z \\ R_4 = y_4(x + y + z + h) \end{cases}$$

Nature des commandes :

Au niveau du module tétraédrique, les commandes sont des impulsions. On justifiera cette nécessité au chapitre 5, lors de l'étude technologique du module.

III-10 FONCTION DE TRANSITION DU MODULE.

Soient $Y = \{1,2,3,4\}$ l ensemble des états du module, $Y' = Y \cup \{p , s\}$ où s et p désignent des éléments fictifs qui tiennent compte des informations en provenance et destination d'un autre module.
$X = \{x,y,z,h\}$ est l'alphabet d'entrée du module.

La fonction de transitions f du module est explicitée ci-dessous :

$$f : Y . X \rightarrow Y'$$
$$f(2,z) = f(3,y) = f(4,x) = f(s,h) = 1$$
$$f(1,x) = f(3,z) = f(4,y) = f(s,h) = 2$$
$$f(1,y) = f(2,x) = f(4,z) = f(s,h) = 3$$
$$f(1,z) = f(2,y) = f(3,x) = f(s,h) = 4$$
$$f(1,h) = f(2,h) = f(3,h) = f(4,h) = p$$

III-11 CONTRAINTES RELATIVES AU DEGRE D'UN ETAT.

Degré d'un état :

On appelle demi-degré intérieur d'un état le nombre de ses prédécesseurs. Le demi-degré extérieur est le nombre de ses successeurs. Le degré d'un état est la somme de ses semi-degrés.

Le degré maximum d'un état représentable par une cellule du module tétraèdrique est huit. Les demi-degrés intérieur et extérieur sont au plus égaux à quatre. Autrement dit, on ne peut avoir au maximum pour 1 état que 3 prédécesseurs et 3 successeurs dans le même module et qu'un seul prédécesseur et successeur dans un module extérieur. Dès que l'un des demi-degrés d'un état est supérieur à quatre, l'extension du degré de la cellule du module représentant cet état s'impose.

Méthode d'extension d'un degré :

L'augmentation du degré d'une cellule s'obtient en multipliant cette cellule. Ainsi, plusieurs cellules peuvent représenter le même état i et, dans le cas du codage un parmi N adopté, les sorties de ces cellules ont la même valeur logique et sont notées y_i ; y_i est le code de l'état i.

a) Cas d'un degré supérieur à huit :

Si l'un des demi-degrés intérieur ou extérieur est supérieur à quatre, les règles de décomposition décrites au chapitre 4 donnent les états dédoublés situés dans des modules différents. Il en est de même du cas où les deux demi-degrés sont supérieurs à quatre.

b) Un autre cas de dédoublement d'états se produit même si chacun des demi-degrés de l'état est inférieur à quatre, quand dans une décomposition graphique, un sommet bien qu'ayant chacun de ses demi-degrés au plus égal à quatre, a au moins deux prédécesseurs ou deux successeurs situés dans un autre sous-graphe (figure 3.9)

Exemple :

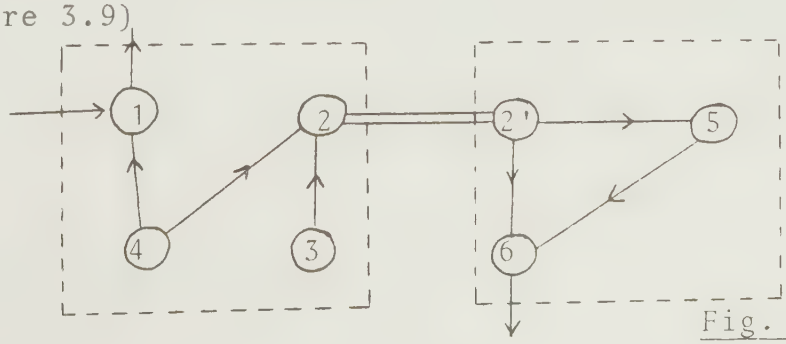


Fig. 3.9

Dans le décomposition de cet exemple, le sommet 2 a deux successeurs situés dans un autre sous-graphe. Le sommet 2 est dédoublé.

Le dédoublement de n mémoires consiste à mettre en parallèle les entrées de ces n cellules et de disposer de 4n directions en entrée et en sortie, ce qui permet à l'état représenté d'avoir 4n prédécesseurs et 4n successeurs.

III-12 BUS DE COMMANDES x,y,z,h.

On sait que les commandes x,y,z, h sont les mêmes pour les 4 cellules d'un module.

Considérons la commande x par exemple :

La transition de l'état 1 à l'état 2 a lieu sous l'effet de x et si x_{1_j} est l'une des 1 variables primaires qui fait passer le système de l'état 1 à l'état 2 :

$$x = \sum_{j=1}^{1} x_{1_j}$$

Mais x exécute aussi les transitions de l'état 2 à 3, de 3 à 4 et de 4 à 1, donc :

$$x = \sum_{i=1}^{4} \sum_{j=1}^{1} x_{i_j}$$

et des expressions analogues pour y, z et h :

$$y = \sum_{i=1}^{4} \sum_{j=1}^{m} y_{i_j}$$

$$z = \sum_{i=1}^{4} \sum_{j=1}^{n} z_{ij}$$

$$h = \sum_{i=1}^{4} \sum_{j=1}^{s} h_{ij}$$

Un cas d'absurdité peut alors se présenter :

Considérons le graphe suivant représenté par un module (figure 3.10) :

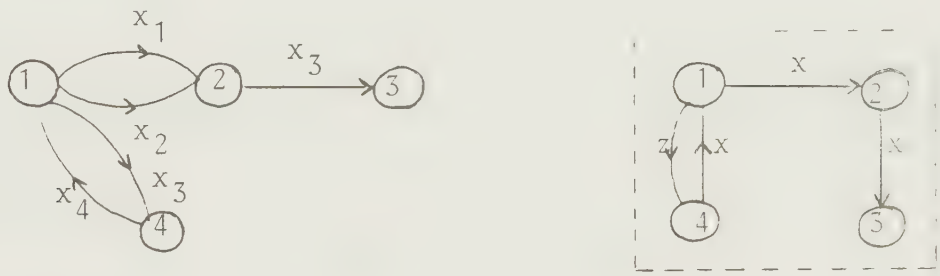


Fig. 3.10

Transitions	Commandes	Variables primaires
1 → 2	x	$x_1 + x_2$
2 → 3	x	x_3
4 → 1	x	x_4
1 → 4	z	x_3

D'après le tableau ci-dessus, on s'aperçoit que le système étant dans l'état 2, l'application de la variable primaire x_3 excite, à la fois la cellule correspondant à l'état 3 et la cellule correspondant à l'état 1. Dans le module correspon-

dant au graphe représenté, on aura :

$$y_3 = 1$$
$$y_1 = 1$$

D'après le codage un parmi N choisi, cela signifie que l'on est à la fois dans l'état 3 et dans l'état 1 ou que l'on a effectué en même temps deux transitions à partir d'un même état, ce qui n'a pas de sens.

Pour écarter cette difficulté, la ième composante du vecteur d'entrée x_i appliquée à la fois à plusieurs commandes d'un même module est remplacée par $y_j x_i$ et $y_k x_i$, y_j et y_k désignant les états du module à partir desquels se font les transitions dues à la variable primaire x_i.

Soit pour l'exemple précédent :

$$x = x_1 + x_2 + x_4 + y_2 x_3$$
$$z = y_1 x_3 .$$

Les bus x, y, z et h sont les mêmes pour tous les modules ne contenant pas d'états dédoublés. En effet :

Considérons l'état dédoublé i ayant 6 successeurs (figure 3.11)

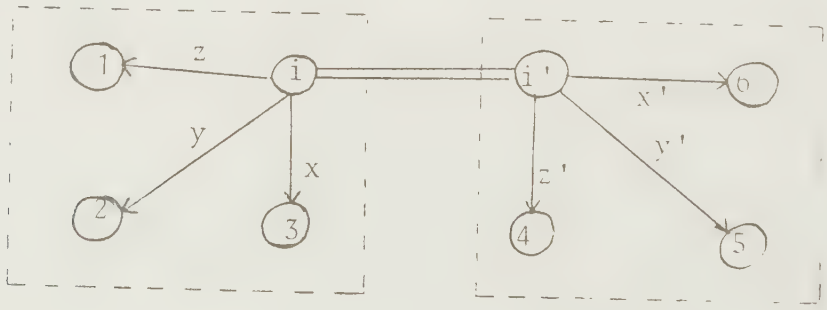


Fig. 3.11

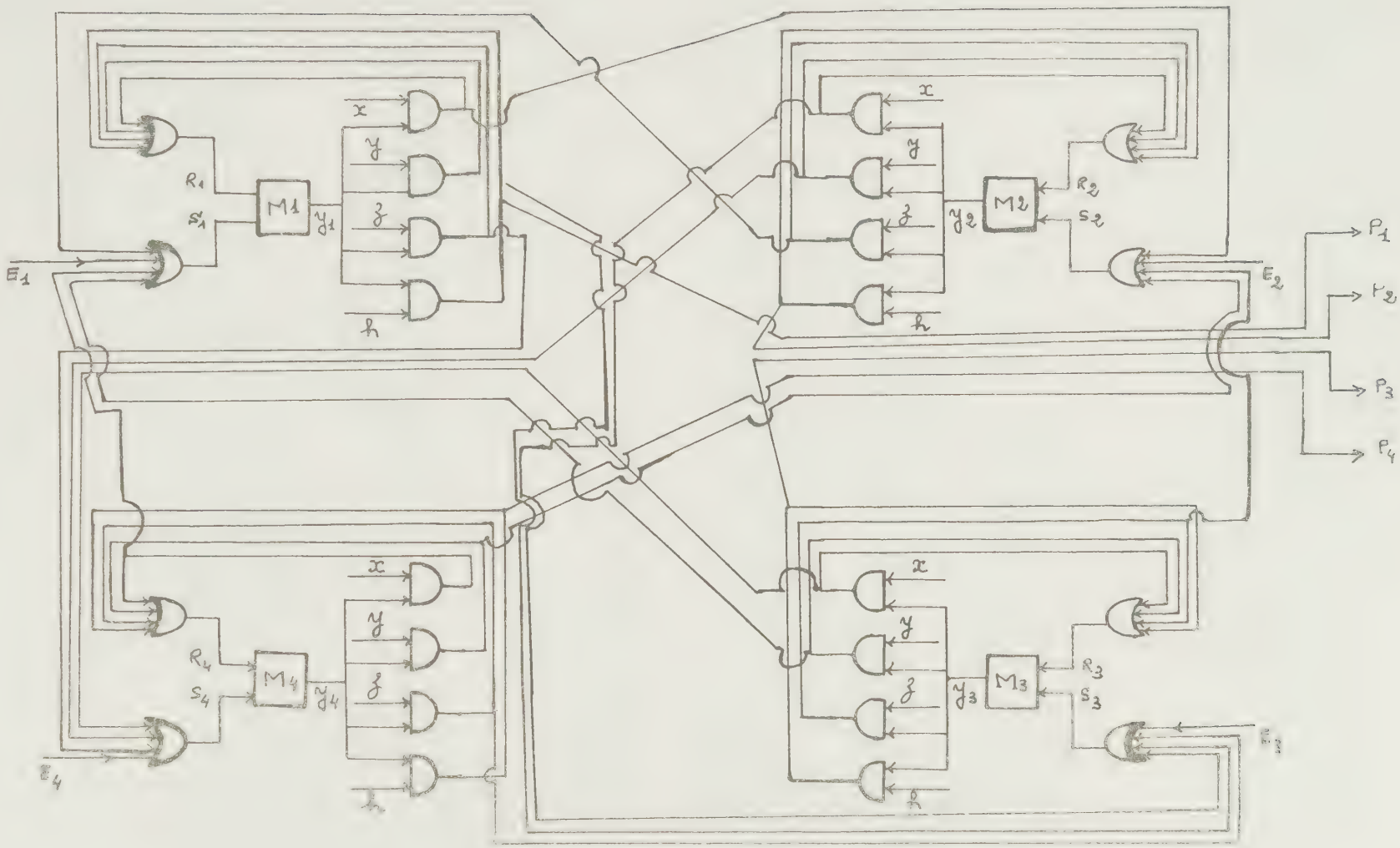


Fig 3.12 : Schéma logique du module électronique

Les cellules i et i' sont à tout instant dans le même état, pour passer dans l'un des 6 états suivants, il faut nécessairement disposer de 6 commandes. C'est pour cette raison que l'on applique des commandes différentes à des modules contenant des états dédoublés

$$x \neq x'$$
$$y \neq y'$$
$$z \neq z'$$
$$h \neq h'$$

III-13 SCHEMA LOGIQUE DU MODULE TETRAEDRIQUE.

Il comprend quatre mémoires M_i (i=1,2,3,4), interconnectées selon la fonction de transitions du paragraphe III.10 (figure 3.12).

III-14 CONCLUSION.

Dans ce chapitre, on a décrit la structure et le fonctionnement du module tétraédrique qui permet de simuler tout automate à 4 états internes. La nature des variables primaires, secondaires et de commande a été précisée. Ce module est de caractère universel et permet la synthèse de tout automate, comme on le verra par la suite.

CHAPITRE IV

APPLICATION A LA SYNTHESE MODULAIRE DES AUTOMATES

IV-1 BUT.

Etant donné un automate A, la possibilité de le décomposer en sous-automates est d'une grande importance. La méthode de Hartmanis basée sur la théorie des partitions permet, quand cela est possible, d'avoir une décomposition d'un automate en sous-automates présentant une certaine indépendance entre eux d'une part, et minimise le nombre des mémoires d'autre part. La dimension de l'automate à décomposer est réduite mais les automates composants ont des tailles variables et ne présentent donc pas un caractère d'universalité.

La décomposition technologique permet une synthèse modulaire à l'aide d'un module simple et unique : le module tétraédrique qui permet de réduire la dépendance entre modules.

IV-2 DECOMPOSITION D'UN AUTOMATE A.

IV.2.1. Méthode.

Soient GA le graphe attaché à un automate A et un sous-graphe SG du graphe GA et qui a, au plus, quatre sommets. Ce sous-graphe est attaché à un automate qui peut être réalisé par un module tétraédrique.

La décomposition de GA en sous-graphes SG permettra de réaliser l'automate A à l'aide de modules tétraédriques. Cette décomposition doit réduire les connections intermodulaires. L'optimisation d'une décomposition graphique est caractérisée par un échange minimum d'informations entre modules. L'optimisation d'une synthèse modulaire permet de minimiser le nombre des liaisons câblées et de localiser, le cas échéant, rapidement un module défectueux. La synthèse modulaire d'un automate de graphe GA nécessite donc :

a) la décomposition de GA en sous-graphes tels que SG,

b) la connexion des modules choisis à l'aide de liaisons intermodulaires.

IV.2.2. *Décomposition non automatique.*

Dans le cas d'un graphe où les liaisons entre sommets sont peu nombreuses, il est souvent facile de trouver des sous-graphes SG par simple examen. Si on prend l'exemple de la figure 4.1, on peut faire deux décompositions en trois modules avec un minimum de 5 liaisons intermodulaires.

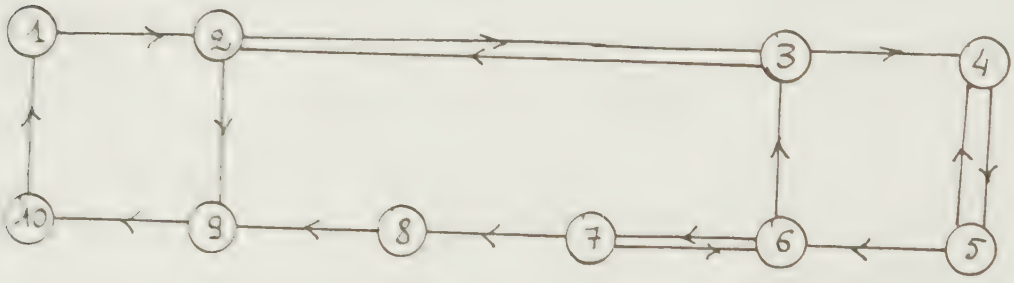


Fig. 4.1

1ère décomposition : 1-2-9-10 ; 3-4-5-6 ; 8-7-0-0
2ème décomposition : 7-8-9-10 ; 1-2-3-4 ; 6-5-0-0 .

Les liaisons intermodulaires correspondant aux deux décompositions sont respectivement 5 et 7, soit respectivement un pourcentage de 33 % et 46 %.

Dans toute décomposition on doit vérifier que chaque état a au plus un prédécesseur et un successeur situé dans un autre module. Si on ne peut trouver une solution, on a un dédoublement simple ou multiple d'un état (cf. § III.11).

IV.2.3. Recherche des sous-graphes SG.

Principe :

La recherche des sous-graphes SG s'effectue en considérant pour chaque sommet la liste de ses précédesseurs et de ses successeurs. Pour un sommet correspondant à un état non dédoublé, il existe au plus un seul prédécesseur et un seul successeur situé dans un module ne contenant pas le sommet considéré.

Pour chacun des quatre sommets constituant un sous-graphe SG, on doit satisfaire la condition précédente. Si pour un sommet donné il n'existe pas de solution ou si l'un de ses demi-degrés est supérieur à quatre, on dédouble ce sommet et on attribue arbitrairement des successeurs et des prédécesseurs aux sommets représentant le même état, tout en respectant la contrainte éventuelle de la dimension d'un demi-degré.

Données définissant l'automate :

Soit GA = (Y,Γ) le graphe attaché à l'automate A à décomposer. On traduit ce graphe sous forme d'une matrice B :

$$B = [(ij)]$$

$$\forall i, \forall j \in Y : (ij) = 1 \iff \exists x \text{ tel que } f(i,x) = j.$$

De même, on utilise la matrice $C = [(kj)]$ pour générer le vecteur de sortie :

$(kj) = 1 \iff$ la kième composante du vecteur de sortie pour l'état j est un.

IV.2.4. Algorithme de composition modulaire d'un automate.

But :

Etant donné le graphe d'un automate, on cherche une décomposition en sous-graphes telle que chacun d'eux soit représentable par le module tétraédrique.

Le nombre des états internes (dimension de la matrice B) et les liaisons entre états sont traduits par la matrice de transition B (§ IV.2.3.).

La première étape du calcul consiste à vérifier que, pour le graphe initial, chaque sommet a au plus 4 prédécesseurs et 4 successeurs. Si cette condition n'est pas remplie, on dédouble le sommet correspondant en répartissant en deux parties égales les liaisons initiales correspondant à l'état dédoublé.

La méthode utilisée dans l'algorithme repose sur la détermination de modules à 4 états au plus avec la contrainte suivante :

Un état d'un module ne peut avoir qu'un seul prédécesseur et un seul successeur situé dans un autre module.

Dans une deuxième étape, on recherche la liste des groupements d'états susceptibles d'être contenus dans le même module en examinant les prédécesseurs et les successeurs de cet état. Ainsi, par exemple pour le graphe de la figure 4.2

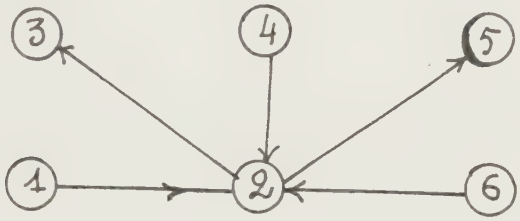


Fig. 4.2

en considérant les prédécesseurs du sommet 2, on peut former la ligne des groupements suivants :

2-1-4-6 ; 2-4-6-0 ; 2-1-6-0 ; 2-1-4-0.

Dans le premier groupement, l'état 2 et ses 3 prédécesseurs sont dans un même module, tandis que dans les 3 autres groupements, il existe un prédécesseur situé dans un autre module ; soient respectivement les sommets 1, 4 et 6. Pour les successeurs, on trouve les groupements suivants :

2-3-5-0 ; 2-3-0-0 ; 2-5-0-0.

Les zéros correspondent aux mémoires non occupées du module à ce niveau de calcul.

Pour n états (y compris les états dédoublés) on obtient n lignes concernant les précédesseurs et n lignes concernant les successeurs. Chaque ligne peut contenir au plus

4 groupements. La comparaison des groupements des lignes des prédécesseurs et des successeurs d'un état permet d'éliminer certains groupements qui ne respectent pas la contrainte imposée.

Supposons que l'on ait les groupements suivants :

sommet 3	:	3-2-4-5	<u>3-4-5-6</u>	<u>3-2-5-6</u>	<u>3-2-4-6</u>
⋮					
sommet 5	:	5-2-0-0			
sommet 6	:	6-8-9-0	6-8-0-0	6-9-0-0	

Considérons le groupement 3-4-5-6, il correspond à un module complet et les deux prédécesseurs du sommet 6 ne peuvent être pris dans ce même module ; donc les groupements soulignés ne peuvent être solutions et sont rejetés. On examine tout sommet d'un groupement que l'on élimine si, pour ce sommet, on a plus d'un successeur ou d'un prédécesseur extérieur au groupement analysé.

Si, après cette opération la jième ligne correspondant à l'état j ne contient aucun groupement, on dédouble l'état j et l'on reprend le processus à la deuxième étape.

Dans la troisième étape, on cherche des solutions à partir des listes des prédécesseurs et des successeurs. Pour cela, à chaque ligne, on attribue un poids égal au nombre de mémoires occupées dans les groupements de cette ligne, dont l'importance est alors chiffrée par la pondération.

Pour calculer les solutions, on forme toutes les décompositions possibles en prenant successivement les lignes des prédécesseurs et des successeurs suivant l'ordre décrois-

sant de leur poids, ce qui permet de réduire les liaisons intermodulaires.

On part d'une solution initiale "vide", c'est-à-dire d'un ensemble de modules dans lesquels aucune mémoire n'est allouée à un état. Les nouvelles solutions s'obtiennent à partir de la ligne de poids le plus fort prise, en intégrant aux solutions initiales les groupements de cette ligne avec les contraintes :

- 4 états au plus par module,
- 1 état n'apparaît qu'une seule fois dans la nouvelle solution.

Le processus est répété jusqu'à ce que tous les états soient pris.

Avec cet algorithme, on choisit donc dans la liste des prédécesseurs et des successeurs la ligne de plus grand poids et établit toutes les solutions possibles ; puis, avec une deuxième ligne, la plus importante parmi celles qui ne sont pas encore examinées, on calcule à partir de ce premier lot de solutions d'autres solutions en respectant les contraintes.

Les solutions qui, vis-à-vis des groupements non encore exploités conduisent à des impossibilités, sont éliminées. De plus, à chaque étape, on supprime les solutions redondantes. Dans le cas où une ligne de la liste des prédécesseurs et des successeurs ne donne aucune solution, on dédouble l'état correspondant à cette ligne et le calcul est repris à l' étape précédente. Après l'exploration de toutes les lignes, on a toutes les décompositions possibles où on choisit celles qui présentent le moins de liaisons intermodulaires.

Exemple : (figure 4.3)

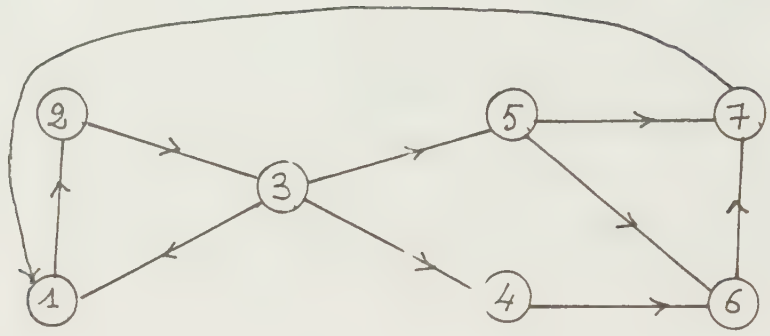


Fig. 4.3

1ère étape : on vérifie que tout demi-degré d'un état $\leqslant$ 4.

2ème étape : groupement des prédécesseurs et successeurs.

sommets 1 : 1-3-7-0 ; 1-3-0-0 ; 1-7-0-0
2 : 2-1-0-0 ; 2-0-0-0
3 : 3-2-0-0 ; 3-0-0-0
4 : 4-3-0-0 ; 4-0-0-0
5 : 5-3-0-0 ; 5-0-0-0
6 : 6-4-5-0 ; 6-4-0-0 ; 6-5-0-0
7 : 7-5-6-0 ; 7-5-0-0 ; 7-6-0-0

sommets 1 : 1-2-0-0 ; 1-0-0-0
2 : 2-3-0-0 ; 2-0-0-0
3 : 3-1-4-5 ; 3-4-5-0 ; 3-1-5-0 ; 3-1-4-0
4 : 4-6-0-0 ; 4-0-0-0
5 : 5-6-7-0 ; 5-6-0-0 ; 5-7-0-0
6 : 6-7-0-0 ; 6-0-0-0
7 : 7-1-0-0 ; 7-0-0-0

3ème étape : on part de la solution vide et on prend la ligne de poids le plus fort parmi les lignes des groupements des

prédécesseurs ou des successeurs, ici la ligne 3s de poids 13. On prend les groupements de cette ligne comme solutions initiales.

Solutions initiales	groupements	nouvelles solutions
Vide	3-1-4-5	3-1-4-5
	3-4-5-0	3-4-5-0
	3-1-5-0	3-1-5-0
	3-1-4-0	3-1-4-0

Le nouveau lot de solutions est :

3-1-4-5 ; 3-4-5-0 ; 3-1-5-0 ; 3-1-4-0

La ligne de poids immédiatement inférieur est : 1p, 6p, 7p.

On prend successivement chaque solution du lot et on examine les groupements de la ligne 1p :

solution initiale	groupements	nouvelles solutions
3-1-4-5	1-3-7-0	néant : 1 et 3 appartiennent à un module complet qui ne peut contenir l'état 7.
	1-3-0-0	1-3-4-5
	1-7-0-0	néant
3-4-5-0	1-3-7-0	néant : l'état 3 doit être dans le même module que 1, 4-5 et 7, ce qui dépasse la capacité d'un module.
	1-3-0-0	3-4-5-1 redondante
	1-7-0-0	1-7-0-0 ; 3-4-5-0

solution initiale	groupements	nouvelles solutions
3-1-5-0	1-3-7-0	1-3-5-7
	1-3-0-0	1-3-5-0
	1-7-0-0	1-3-5-7 redondante
3-1-4-0	1-3-7-0	1-3-4-7
	1-3-0-0	1-3-4-0
	1-7-0-0	1-3-4-7 redondante

Nouveau lot de solutions :

3-1-4-5 ; 1-3-5-7 ; 1-3-4-7 ; 1-3-5-0 ; 1-3-4-0 ; 3-4-5-0 ; 1-7-0-0

On prend chacune des solutions précédentes comme solutions initiales et on répète le même processus avec les lignes pas encore prises, dans l'ordre décroissant de leur poids.

solution initiale	groupements	nouvelles solutions
3-1-4-5	6-5-4-0	néant
	6-4-0-0	néant
	6-5-0-0	néant
1-3-5-7	6-5-4-0	néant
	6-4-0-0	1-3-5-7 ; 6-4-0-0
	6-5-0-0	néant
1-3-4-7	6-5-4-0	néant
	6-4-0-0	néant
	6-5-0-0	1-3-4-7 ; 6-5-0-0
1-3-5-0	6-5-4-0	néant
	6-4-0-0	1-3-5-0 ; 6-4-0-0
	6-5-0-0	1-3-5-6

1-3-4-0	6-5-4-0	néant
	6-4-0-0	1-3-4-6
	6-5-0-0	1-3-4-0 ; 6-5-0-0
3-4-5-0 ; 1-7-0-0	6-5-4-0	3-4-5-6 ; 1-7-0-0
	6-4-0-0	3-4-5-6 ; 1-7-0-0
	6-5-0-0	3-4-5-6 ; 1-7-0-0

Le nouveau lot de solutions après l'examen de cette ligne est :

1-3-5-7 ; 6-4-0-0 ; 1-3-4-7 ; 6-5-0-0 ; 1-3-5-0 ; 6-4-0-0 ;
1-3-4-6 ; 1-3-4-0 ; 6-5-0-0 ; 3-4-5-6 ; 1-7-0-0.

<u>Recherche de nouvelles solutions</u> :

solution initiale	groupements	nouvelles solutions
1-3-5-7 ; 6-4-0-0	7-6-5-0	néant
	7-6-0-0	néant
	7-5-0-0	1-3-5-7 ; 6-4-0-0
1-3-4-7 ; 6-5-0-0	7-6-5-0	néant
	7-6-0-0	néant
	7-5-0-0	néant
1-3-5-0 ; 6-4-0-0	7-6-5-0	néant
	7-6-0-0	1-3-5-0 ; 7-6-4-0
	7-5-0-0	1-3-5-7 ; 6-4-0-0
1-3-4-6	7-6-5-0	néant
	7-6-0-0	néant
	7-5-0-0	1-3-4-6 ; 7-5-0-0
1-3-4-0 ; 6-5-0-0	7-6-5-0	1-3-4-0 ; 5-6-7-0
	7-6-0-0	1-3-4-0 ; 5-6-7-0
	7-5-0-0	1-3-4-0 ; 5-6-7-0

3-4-5-6 ; 1-7-0-0	7-6-5-0	néant
	7-6-0-0	néant
	7-5-0-0	néant

Nouveau lot de solutions :

1-3-5-7 ; 6-4-0-0 ; 1-3-5-0 ; 7-6-4-0 ; 1-3-4-6 ; 7-5-0-0 ; 1-3-4-0 ; 5-6-7-0.

Recherche de nouvelles solutions :

solution initiale	groupements	nouvelles solutions
1-3-5-7 ; 6-4-0-0	5-6-7-0	néant
	5-6-0-0	néant
	5-7-0-0	1-3-5-7 ; 6-4-0-0
1-3-5-0 ; 7-6-4-0	5-6-7-0	néant
	5-6-0-0	néant
	5-7-0-0	néant
1-3-4-6 ; 7-5-0-0	5-6-7-0	néant
	5-6-0-0	néant
	5-7-0-0	1-3-4-6 ; 7-5-0-0
1-3-4-0 ; 5-6-7-0	5-6-7-0	1-3-4-0 ; 5-6-7-0
	5-6-0-0	1-3-4-0 ; 5-6-7-0
	5-7-0-0	1-3-4-0 ; 5-6-7-0

Nouveau lot de solutions

1-3-4-6 ; 7-5-0-0 ; 1-3-4-0 ; 5-6-7-0 ; 1-3-5-7 ; 6-4-0-0.

solution initiale	groupements	nouvelles solutions
1-3-4-6 ; 7-5-0-0	2-1-0-0	néant
	2-0-0-0	1-3-4-6 ; 7-5-2-0
1-3-4-0 ; 5-6-7-0	2-1-0-0	1-2-3-4 ; 5-6-7-0
	2-0-0-0	1-3-4-0 ; 5-6-7-2
1-3-5-7 ; 6-4	2-1-0-0	néant
	2-0-0-0	1-3-5-7 ; 6-4-2-0

Nouveau lot de solutions :

1-3-4-6 ; 2-5-7-0 ; 1-2-3-4 ; 5-6-7-0 : 1-3-4-0 ; 5-6-7-2 ; 1-3-5-7 ; 6-4-2-0.

<u>Recherche de nouvelles solutions</u> :

solution initiale	groupements	nouvelles solutions
1-3-4-6 ; 2-5-7-0	2-1-0-0	néant
	2-0-0-0	1-3-4-6 ; 2-5-7-0
1-2-3-4 ; 5-6-7-0	2-1-0-0	1-2-3-4 ; 5-6-7-0
	2-0-0-0	1-2-3-4 ; 5-6-7-0
1-3-4-0 ; 5-6-7-2	2-1-0-0	néant
	2-0-0-0	1-3-4-0 ; 2-5-6-7
1-3-5-7 ; 6-4-2-0	2-1-0-0	néant
	2-0-0-0	1-3-5-7 ; 2-4-6-0

Nouveau lot de solutions :

1-3-4-6 ; 2-5-7-0 ; 1-2-3-4 ; 5-6-7-0 ; 1-3-4-0 ; 2-5-6-7 ; 1-3-5-7 ; 2-4-6-0.

Le lot de solutions n'a pas changé. De même, en considérant toutes les lignes non encore exploitées, on trouve les mêmes solutions.

La décomposition de l'automate défini par le graphe de la figure 4.3 est :

1) 1-3-4-6 ; 2-5-7-0 nombre de liaisons intermodulaires : 6
2) 1-2-3-4 ; 5-6-7-0 nombre de liaisons intermodulaires : 3
3) 1-3-4-0 ; 2-5-6-7 nombre de liaisons intermodulaires : 5
4) 1-3-5-7 ; 2-4-6-0 nombre de liaisons intermodulaires : 6

La solution avec le moins de liaisons intermodulaires est :

1-2-3-4 ; 5-6-7-0.

La figure 4.4 donne le schéma de décomposition de l'automate.

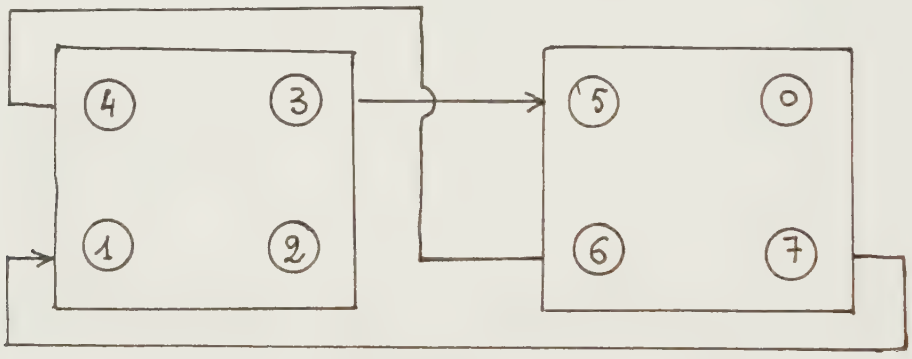


Fig. 4.4
Schéma de décomposition de l'automate

CONCLUSION.

L'algorithme donne des solutions ; parmi ces solutions, on prend celles qui donnent le moins de liaisons à câbler.

IV-3 GENERATION DU VECTEUR DE SORTIE.

Les équations de récurrence définissant un automate sont :

$$Y_{n+1} = F(X_n , Y_n) \qquad (1)$$

$$Z_n = G(X_n , Y_n) \qquad (2)$$

où la première est utilisée pour effectuer la décomposition de l'automate, tandis que la deuxième donne la sortie du système.

Pour générer simplement le vecteur de sortie, dans le cas d'un codage un parmi n, on se ramène à un modèle de Moore et chaque composante z_i du vecteur Z_n est obtenue en faisant la somme des variables secondaires y_j correspondant aux états j pour lesquels cette composante vaut un. On obtient :

$$z_i = \sum_j y_j \qquad (3)$$

En reconsidérant les équations (1) et (2) précédentes, on obtient :

$$Y'_{n+1} = F(X_n , Y'_n)$$

$$Z_n = G(Y'_n)$$

A partir de la matrice $C = [(kj)]$ du chapitre IV.2.3, on détermine tous les y_j relatifs à chaque composante z_i et un opérateur effectue la somme (3) correspondante.

CHAPITRE V
REALISATION-UTILISATION

V-1 REALISATION.

V.1.1. *Technologie employée.*

Les éléments employés dans la construction du module sont des circuits intégrés de la logique TTL. D'autres composants électroniques conventionnels ou pneumatiques sont utilisables.

V.1.2. *Nature des signaux.*

Le codage adopté est 1 parmi n, à chaque état i de l'automate est associé une mémoire dont la sortie y_i vaut un si et seulement si l'automate est dans l'état i. Comme on l'a vu au § III.7, le passage d'un état i à un état j sous l'action d'une commande x,y,z ou h, nécessite les deux opérations suivantes :

- mise à un de la mémoire j : $y_j = 1$
- remise à zéro de la mémoire i : $y_i = 0$

Une partie de la figure 3.9 est reproduite en figure 5.1, où on examine le passage de l'état 1 à l'état 2 sous l'action de la commande x :

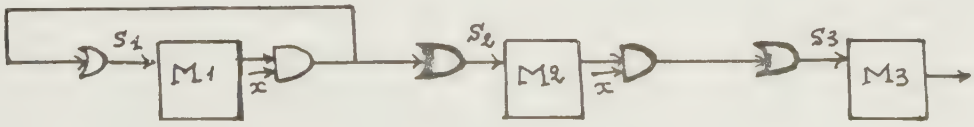


Fig. 5.1

Etat initial : $y_1 = 1$
$y_2 = y_3 = 0$

Lors de l'application de x :

$S_2 = 1$, M_2 bascule et si x est un niveau ou une impulsion trop large, S_3 devient haut, M_3 bascule, etc... et l'on a un cycle continu : M_1 M_2 M_3 M_4 M_1 ...

On voit donc qu'il est nécessaire d'avoir une commande impulsionnelle afin d'éviter le parcours de ce cycle. De plus cette impulsion doit retomber à zéro dès le premier basculement afin d'éviter la propagation indésirable de y_i vers d'autres états. La réalisation d'une impulsion très fine conduit à des résultats aléatoires compte-tenu de la diversité des temps de réponse de circuits commerciaux.

Pour pallier ces inconvénients, on effectue l'opération en deux temps :

1°) élaboration du signal S_2,

2°) puis prise en compte de ce signal S_2 pour faire basculer M_2 en utilisant un signal très bref, par exemple le front montant d'une impulsion et en même temps remettre la bascule M_1 à zéro.

V.1.3. Fonctionnement du module.

V.1.3.1. Transition entre états appartenant au même module :

Chaque mémoire M_i est une bascule D de la logique TTL, d'entrée D_i et de sortie y_i. Les instants n et n+1 sont relatifs à la transition du signal T de 0 à 1, on a :

$$y_i(n+1) = D_i(n).$$

Le transfert de l'entrée D_i vers la sortie y_i a lieu au front montant de l'impulsion T.

$D_i(n)$ est l'état de l'entrée de la bascule avant la transition de 0 à 1 et $y_i(n+1)$ est l'état de la sortie après cette transition de T.

Les différentes entrées D_i du module ont pour expressions :

$$D_1 = xy_4 + yy_3 + zy_2 + hE_1 = S_1$$
$$D_2 = xy_1 + yy_4 + zy_3 + hE_2 = S_2$$
$$D_3 = xy_2 + yy_1 + zy_4 + hE_3 = S_3$$
$$D_4 = xy_3 + yy_2 + zy_1 + hE_4 = S_4$$

x,y,z et h sont des commandes impulsionnelles ou entrées primaires du module.

y_i et E_i sont des variables secondaires : $y_i(i=1,2,3,4)$ appartiennent au même module ; $E_i(i=1,2,3,4)$ appartiennent à des modules quelconques. Ces variables sont des niveaux.

Les signaux $D_i(i=1,...,4)$ sont présents à l'entrée d'une bascule aussitôt dès l'application d'une commande. Le passage au niveau haut d'un état j et la remise à zéro de l'état i prédécesseur de j quand une commande x,y,z ou h est appliquée à l'état i doit se faire avec un retard suffisant pour que les signaux D_i aient eu le temps de se positionner. Le changement d'état se fait alors pour une cellule M_i sur le front montant du signal T tel que :

$$T = (x + y + z + h)\ e^{-j\theta}$$

En l'absence de toute commande :

$$D_k = 0 \qquad k = 1, 2, 3, 4.$$

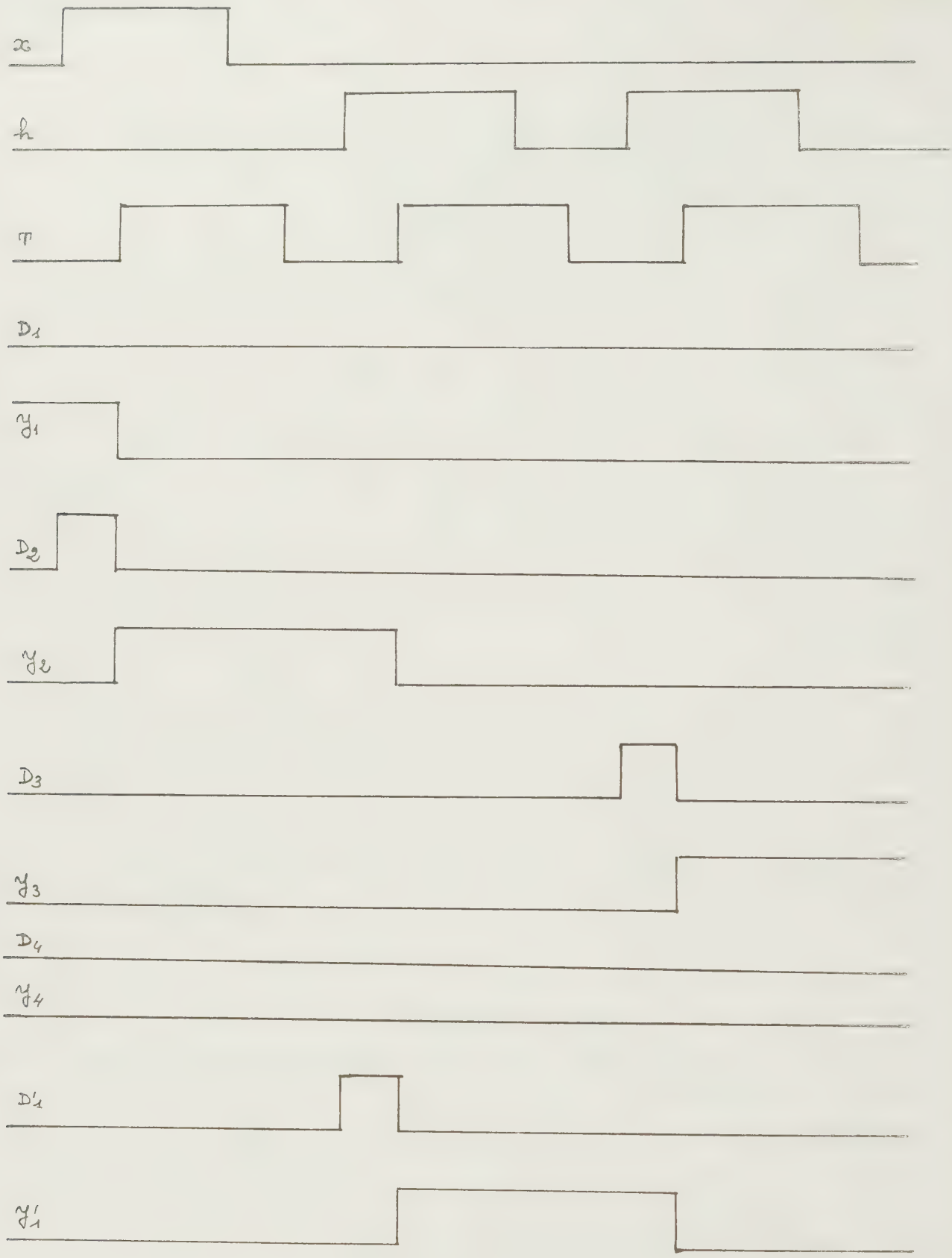


fig 5-2. Diagramme des transitions ; y_1, y_2, y_3 sont situés dans un même module. y'_1 appartient à un module différent. Le diagramme est relatif aux transitions $y_1 \to y_2 \to y'_1 \to y_3$

Dès l'arrivée d'une commande x, y, z ou h qui fait la transition de l'état i à l'état j, seul D_j devient haut ($D_j = 1$).

La remise à zéro se fait par la même commande qui transfert les entrées D_k aux sorties y_k des mémoires. Un seul signal D_i est suffisant pour jouer le rôle de S_i et R_i (§III8). Le signal T est commun à tous les modules et apparaît à chaque changement du vecteur d'entrée : c'est la somme retardée de toutes les commandes.

Pour un vecteur d'entrée à n composantes $e_i (i=1,\ldots,n)$ correspondent 2n commandes x_i et x_i' relatives aux transitions de 0 à 1 et de 1 à 0 de e_i. On a donc :

$$T = \sum_{i=1}^{n} (x_i + x_i') \, e^{-j\theta} .$$

V.1.3.2. Diagramme temporel des transitions à l'intérieur d'un même module et entre modules :

La figure 5.2 montre les transitions de l'état 1 à 2, de 2 à 1'et de 1' à 3. Les états 1,2,3 sont dans un même module ; l'état 1' est dans un module différent du premier. Considérons successivement ces transitions :

Transition de l'état 1 à l'état 2 situés dans un même module :

Initialement :

$$y_1 = 1$$

$$x = y = z = h = T = D_1 = 0$$

Le passage de 1 à 2 est effectué sous la commande x.

$x = 1$

$D_1 = 0 \quad (xy_4 = 0)$

$D_2 = xy_1 + yy_4 + zy_3 + hE_2 = 1 \quad (xy_1 = 1)$

$T = (x+y+z+h)\ e^{-j\theta} = 0$

Quand T passe à 1 après le retard θ relatif à x, en même temps y_1 et y_2 passent respectivement à 0 et 1 (transfert de D_1 et D_2 vers les sorties des mémoires y_1 et y_2) et ensuite D_2 devient nulle (car y_1 devient nul).

De même pour la transition de 2 à 1' on a :

$E_1' = y_2 = 1$

$D_1' = h\ E_1'$

$T = h\ e^{-j\theta}$.

Puis, la transition de l'état 2 à l'état 1' a lieu à l'instant - après l'application de h.

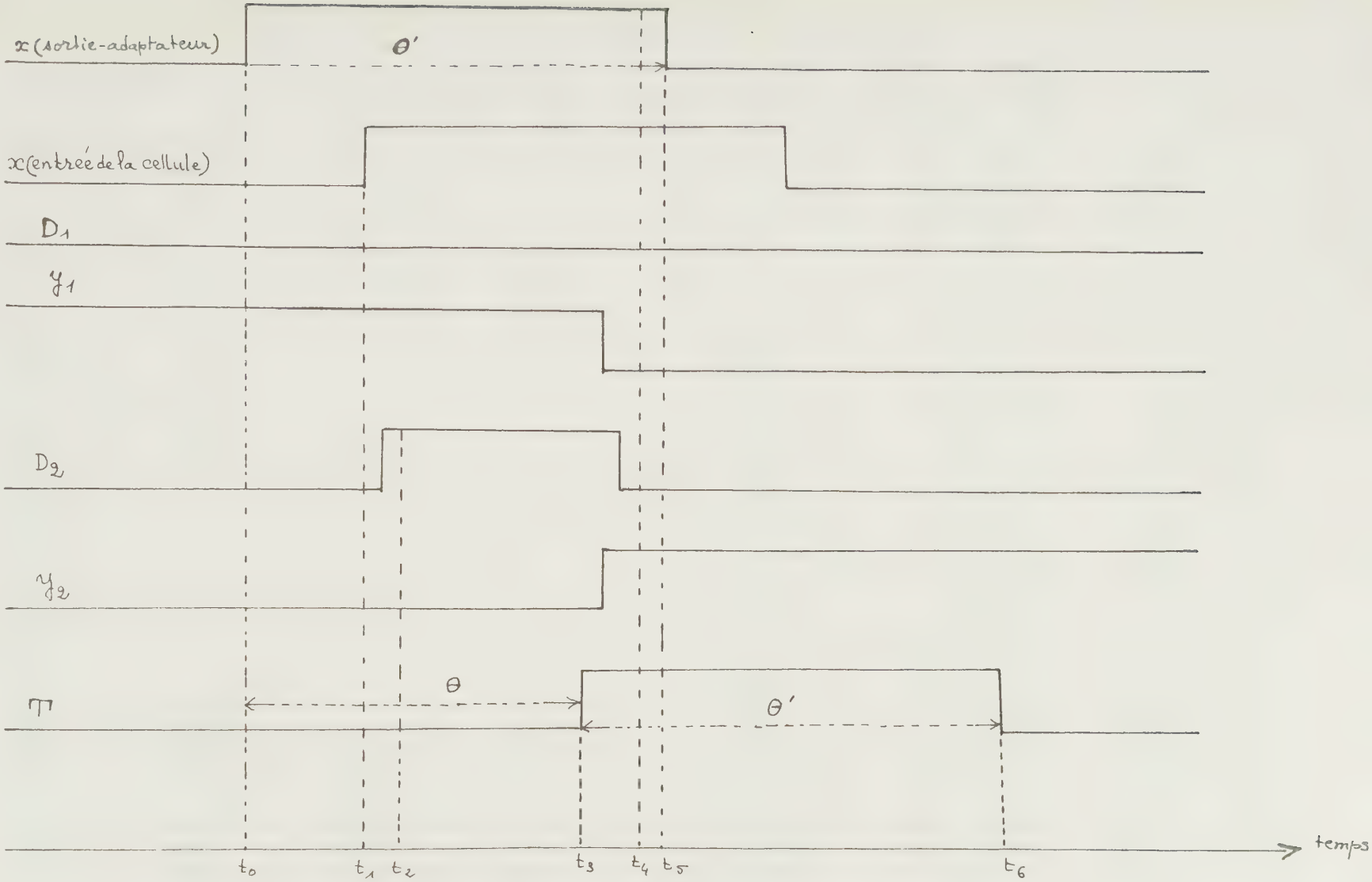
La transition de 1' à 3 s'effectue de façon analogue :

$E_3 = y_1' = 1$

$D_3 = E_2$

$T = h\ e^{-j\theta}$.

Ainsi, on peut effectuer toutes les transitions voulues à l'intérieur d'un même module ou entre modules.



$2\tau = t_1 - t_0$; $\tau_s = t_2 - t_1$; $\tau_h = t_5 - t_4$; $\theta' = t_6 - t_3$

L'instant d'arrivée de T doit être compris entre t_2 et t_4 ; $\theta' \geqslant \theta'_{min} = 66$ ns

fig 5-3 - Diagramme des phases d'une transition -

V.1.3.3. Commandes et réponse du module :

Analysons le diagramme des phases de la figure 5.3 représentant la transition de l'état 1 à l'état 2 sous la commande x.

On voit sur ce diagramme que les entrées D_1 et D_2 se retrouvent sur les sorties y_1 et y_2 dès que T passe au niveau haut. Le signal que l'on veut transférer au front montant de T doit être présent au moins τ_s et τ_h avant et après T.

D'après la réalisation, une commande quelconque traverse deux portes TTL pour donner un signal D_i ; soit 2τ la durée de cette traversée. La largeur minimale de l'impulsion T est θ'_{min} (= 66ns dans notre cas). C'est donc la plus petite valeur de θ'. On a alors θ' désignant la largeur d'une commande :

$$\theta > 2\tau + \tau_s$$

$$2\tau + \theta' > \theta + \tau_h$$

Soit : $$2\tau + \tau_s < \theta < \theta' - \tau_h + 2\tau$$

Valeurs numériques :

$$\theta' > \theta'_{min} \simeq 66ns$$

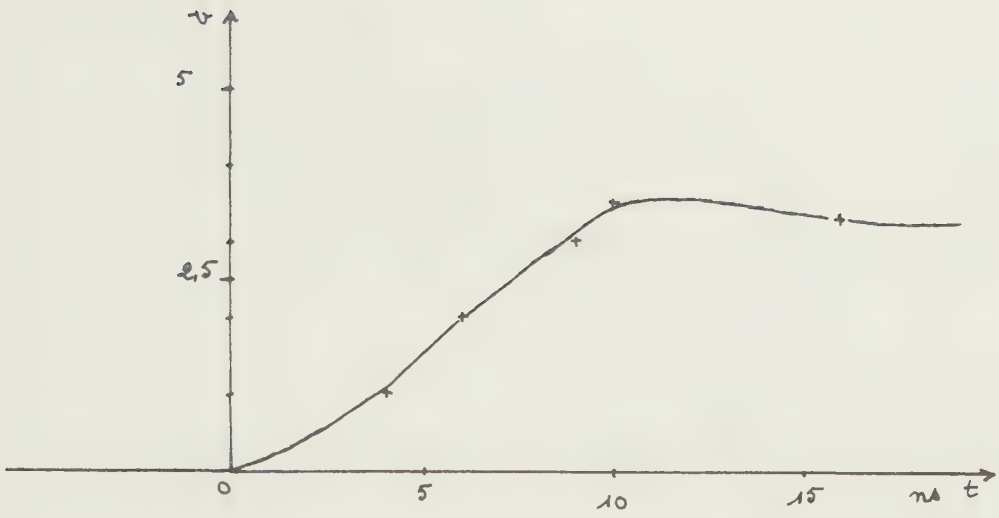
$$\tau \simeq 30ns$$

$$\tau_s \simeq 20ns$$

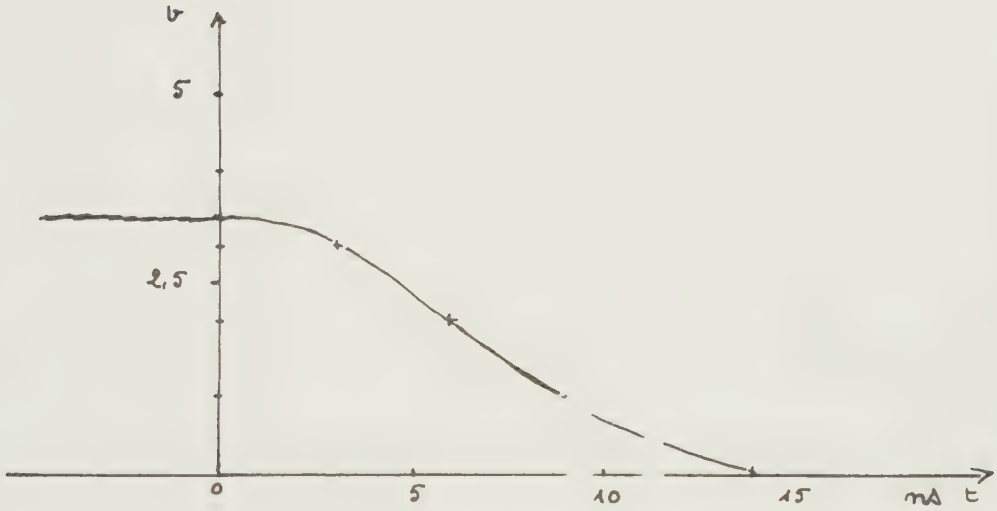
$$\tau_h \simeq 3ns$$

Le plus petit intervalle de variation de θ est :

$$80ns < \theta < \theta'_{min} + 2\tau - \tau_h = 125ns.$$

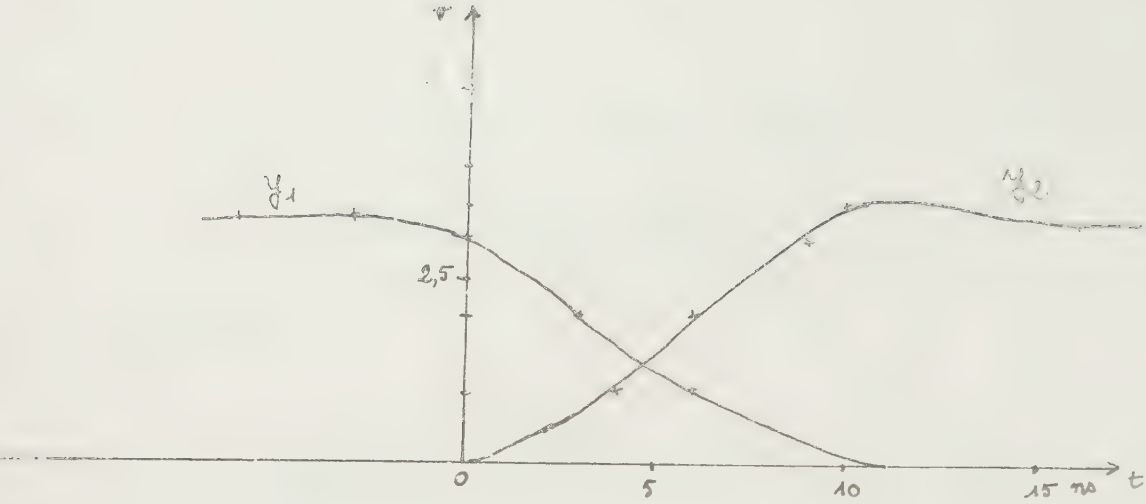


Transition vers l'état haut d'une mémoire.

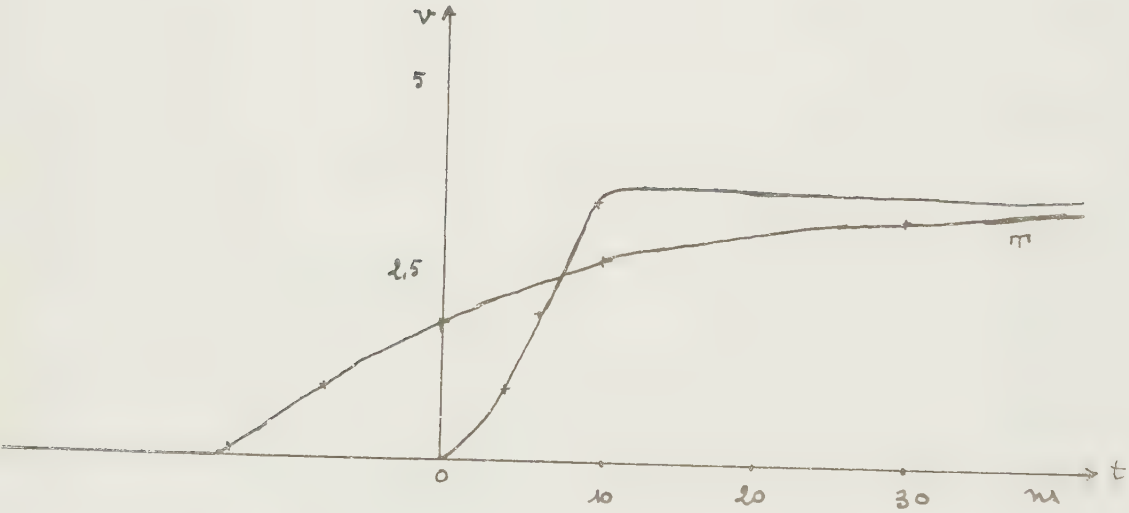


Transition vers le niveau 0 d'une cellule.

fig 5-4



Transition entre deux états voisins du module.



Réponse d'une mémoire et signal T

fig 5-4.

Le temps de réponse du module est le temps nécessaire à la propagation d'un niveau de l'entrée vers la sortie. Sa valeur est la somme des temps de réponse des éléments traversés par le signal considéré. Pour une mémoire, on a relevé (cf. Fig. 5.4)

$$\tau_m = 10ns.$$

Temps de réponse globale t_s d'une cellule : c'est le temps nécessaire à l'apparition de la sortie dès qu'une commande est appliquée.

V.1.3.4. Forme d'onde des différents signaux :

La figure 5.4 montre les réponses obtenues.

V.1.3.5. Adaptateur de commandes :

L'adaptateur doit répondre à la contrainte imposée par la mémoire utilisée : transformer une variation du vecteur d'entrée en une impulsion θ' de durée déterminée.

a) Description :

La dimension du vecteur d'entrée est variable. L'adaptateur est conçu pour une dimension maximale de 8 composantes. La mise en parallèle de plusieurs adaptateurs permet de disposer d'un vecteur d'entrée de dimension quelconque.

L'adaptateur comprend 3 parties :

- un registre à 8 bits pour stocker le vecteur d'entrée,
- un comparateur pour détecter les transitions entre deux valeurs successives du vecteur d'entrée,

- un bloc pour générer une impulsion x_j ou x'_j selon que la composante e_j du vecteur d'entrée passe de 0 à 1 ou de 1 à 0.

Le schéma général de l'adaptateur est donné sur la figure 5.5.

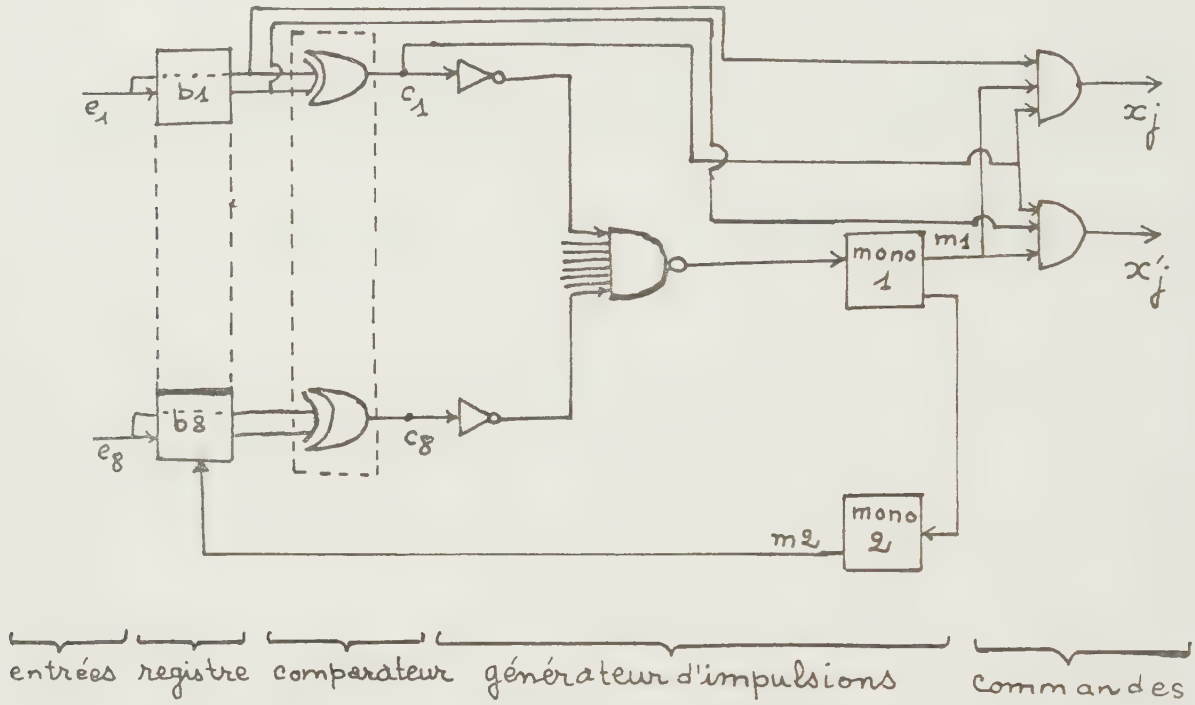


Fig. 5.5
Adaptateur de commandes

b) Fonctionnement :

A l'état permanent, $c_1 = c_2 = \ldots c_8 = 0$ et $m_1 = m_2 = 0$ (fig. 5.5) ; s'il y a une variation du vecteur d'entrée, par exemple la composante e_1, $c_1 = 1$ et la sortie du NAND à 8 entrées passe au niveau haut, ce qui déclenche une impulsion en m_1 et à la fin de cette impulsion, le monostable 2 produit à son tour une impulsion m_2 qui provoque le transfert du nouveau vecteur d'entrée dans le registre à 8 bits.

V.1.3.6. *Calibrage des impulsions données par l'adaptateur : la largeur minimale de θ' est :*

$$\theta' > \theta'_{min} = 66 \text{ ns.}$$

La borne supérieure de cette impulsion est quelconque ; toutefois, tant que l'impulsion est haute, le vecteur d'entrée ne doit pas changer, sinon la transition d'un état à un autre sous cette commande ne sera pas effectuée. Deux variations successives du vecteur d'entrée doivent être distantes dans le temps d'au moins $\max(t_s, \theta')$.

V.1.3.7. *Entrées-sorties de l'adaptateur de commandes :*

Le nombre des composantes du vecteur d'entrée est 8. Un vecteur d'entrée à 8 composantes est suffisant pour simuler des systèmes séquentiels de taille moyenne. L'emploi de plusieurs adaptateurs en parallèle permet de composer tout vecteur d'entrée. Dans le cas où les entrées sont des niveaux, pour éviter les rebondissements lors de la variation de la position d'un interrupteur mécanique, on utilise un registre dans lequel on met le vecteur d'entrée quand les rebondissements sont terminés, en appliquant une impulsion de transfert. Le contenu de ce registre est l'entrée du simulateur.

Les sorties de l'adaptateur sont les 16 impulsions calibrées (§ précédent) x_i, x'_i ($i = 1, 2, \ldots, 8$).

V.1.3.8. *Simulateur :*

Le simulateur permet de réaliser des systèmes séquentiels dont le nombre d'états est au plus égal à vingt et dont le vecteur de sortie a au plus cinq composantes et un alphabet d'entrée de huit éléments maximum.

Il comprend :

- un registre à 8 bits stockant le vecteur d'entrée,
- un adaptateur de commandes,
- 5 modules tétraédriques,
- un générateur de vecteur de sortie,
- une plaquette d'opérateur NAND.

V-2 UTILISATION.

La simulation des automates définis par le graphe se fait par interconnection des modules tétraédriques et génération du vecteur de sortie. Les états représentés par un module sont donnés par le programme de décomposition. Le câblage se fait comme indiqué au § III.12.

Exemple : Soit à réaliser l'automate dont le graphe des états internes est donné en figure 5.6.

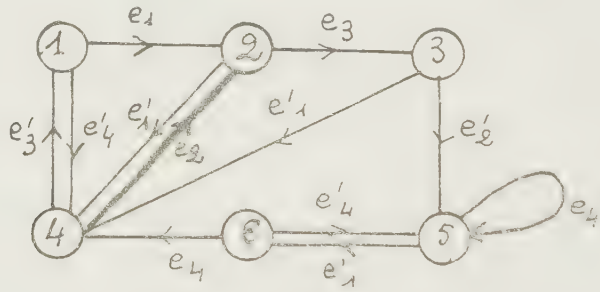


Fig. 5.6

Le module MD_1 contient les états 1 2 3 4.
Le module MD_2 contient les états 6 et 5.

Il n'y a pas d'état dédoublé. Les bus des deux modules peuvent être les mêmes.

a) Commandes :

$$x = e_1 + e'_1 + e_3 + e'_3$$

$$y = e_2 + e'_1$$

$$z = e'_4 + e_4$$

$$h = e_4 + e'_2$$

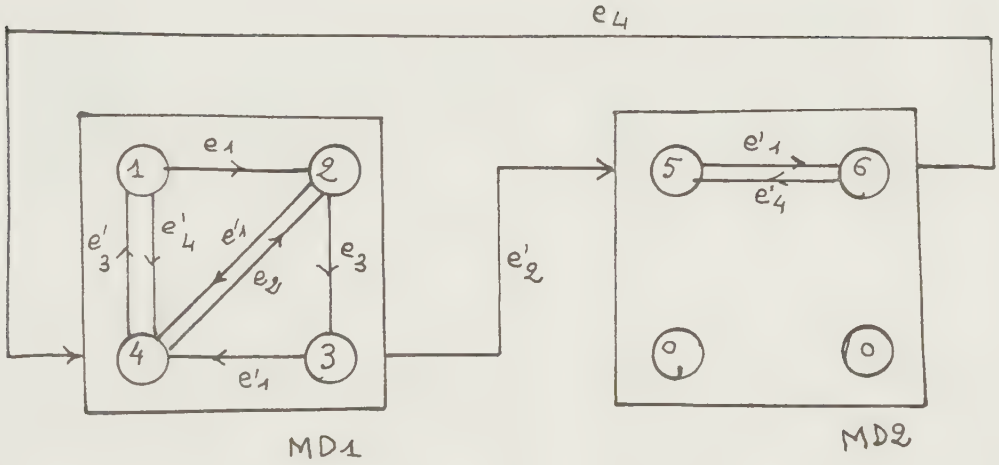


Fig. 5.7

On fait alors la somme des différentes composantes du vecteur d'entrée que l'on applique aux commandes correspondantes.

b) Interconnexion entre modules :

Si les états i et j sont situés dans deux modules différents, et que la relation de transition Γ :

$$i \Gamma j \quad \text{est vraie,}$$

on relie la sortie secondaire p_i à l'entrée secondaire E_j (cf. schéma 3.12).

Dans l'exemple traité, on relie la sortie P_i de la cellule contenant l'état 3 à l'entrée E_j de la cellule j contenant l'état 5 ; de même, on relie la sortie de la cellule représentant l'état 6 à l'entrée de la cellule correspondant à l'état 4. On obtient le schéma de la figure 5.7.

c) Sortie :

La sortie est générée conformément au § IV.3 Comme on dispose sur le module de $\bar{y}_j$, on a pour une composante z_k du vecteur de sortie :

$$z_k = \sum_j y_j = \overline{\pi \bar{y}_j}$$

Exemple traité par calculateur (cf. annexe).

CONCLUSION.

La méthode de décomposition d'un automate basée sur les partitions est lourde à mettre en œuvre quand le nombre de variables est grand puisqu'on travaille sur les deux ensembles X et Y, et donne des sous-automates de tailles variables.

Pour éviter ces inconvénients, nous avons associé au fonctionnement d'un automate un graphe décomposable en sous-graphes modules.La décomposition technologique correspondante est simple puisqu'on ne traîte que les éléments du seul ensemble Y, les sommets du graphe. Un algorithme de décomposition donne des solutions avec minimisation des liaisons intermodulaires.

Nous avons considéré plus particulièrement le module tétraédrique qui peut simuler tout automate de sous-graphe SG

et qui :

- est universel,
- est programmable et possède un nombre minimum de 4 commandes pour les 4 cellules qui le composent,
- est intégrable à moyenne ou à grande échelle.

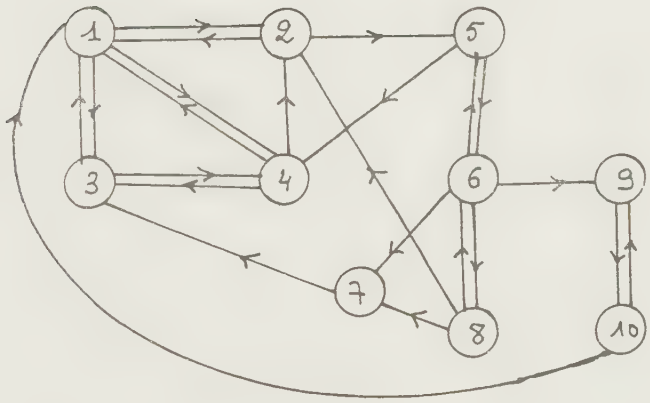
La synthèse d'un automate synchrone ou asynchrone à l'aide du module tétraédrique se fait alors par simple câblage de liaisons intermodulaires réduites et génération du vecteur de sortie.

A N N E X E

Exemples de décomposition automatique.

Soit à décomposer les automates dont les graphes suivent :

Exemple A



TRANSITIONS

0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	0	1	0	1	1	1	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	0

SUCCESSEURS ET PREDECESSEURS

[illegible]	1	2	3	4	1	3	4	0	1	2	4	0	1	2	3	0	[illegible]
2	2	1	5	0	2	5	0	0	2	1	0	0	0	0	0	0	[illegible]
[illegible]	3	1	4	0	3	4	0	0	3	1	0	0	0	0	0	0	[illegible]
[illegible]	4	1	2	3	4	2	3	0	4	1	3	0	4	1	2	0	[illegible]
3	5	4	6	0	5	6	0	0	5	4	0	0	0	0	0	0	[illegible]
[illegible]	6	7	8	9	6	5	8	9	6	5	7	9	6	5	7	6	[illegible]
[illegible]	7	3	0	0	7	0	0	0	0	0	0	0	0	0	0	0	[illegible]
[illegible]	8	2	6	7	8	6	7	0	8	2	7	0	8	2	[illegible]	0	13
[illegible]	9	10	0	0	9	0	0	0	0	0	0	0	0	0	0	0	[illegible]
3	10	1	9	0	10	9	0	0	10	1	0	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	3	4	10	1	2	4	10	1	2	3	10	1	2	3	4	16
4	2	1	4	8	2	4	8	0	2	1	8	0	2	1	4	0	13
4	3	1	4	7	3	4	7	0	3	1	7	0	3	1	4	0	13
4	4	1	3	5	4	3	5	0	4	1	5	0	4	1	3	0	13
3	5	2	6	0	5	6	0	0	5	2	0	0	0	0	0	0	7
3	6	5	8	0	6	8	0	0	6	5	0	0	0	0	0	0	7
3	7	6	8	0	7	8	0	0	7	6	0	0	0	0	0	0	7
2	8	6	0	0	8	0	0	0	0	0	0	0	0	0	0	0	3
3	9	6	10	0	9	10	0	0	9	6	0	0	0	0	0	0	7
2	10	9	0	0	10	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

nombre de groupements

poids p(i) de la i^{ème} ligne

SOLUTION

6	7	8	9	1	3	4	10
6	5	8	9	1	3	4	10
6	5	7	9	1	3	4	10
6	5	7	8	1	3	4	10
6	7	8	9	1	2	4	10
6	5	8	9	1	2	4	10
6	5	7	9	1	2	4	10
6	5	7	8	1	2	4	10
6	7	8	9	1	2	3	10
6	5	8	9	1	2	3	10
6	5	7	9	1	2	3	10
6	5	7	8	1	2	3	10
6	7	8	9	1	2	3	4
6	5	8	9	1	2	3	4
6	5	7	9	1	2	3	4
6	5	7	8	1	2	3	4

SOLUTION

6	7	8	9	1	2	3	4
6	5	8	9	1	2	3	4
6	5	7	9	1	2	3	4
6	5	7	8	1	2	3	4
6	7	8	9	1	3	4	10
6	5	8	9	1	3	4	10
6	5	7	9	1	3	4	10
6	5	7	8	1	3	4	10
6	7	8	9	1	2	4	10
6	5	8	9	1	2	4	10
6	5	7	9	1	2	4	10
6	5	7	8	1	2	4	10
6	7	8	9	1	2	3	10
6	5	8	9	1	2	3	10
6	5	7	9	1	2	3	10
6	5	7	8	1	2	3	10

SOLUTION

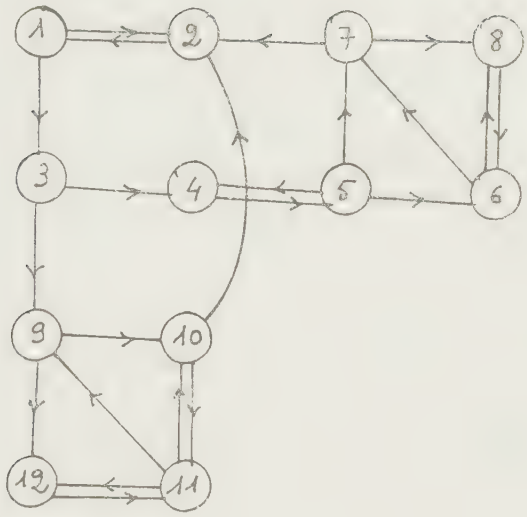
6	7	8	9	1	2	3	4
6	5	8	9	1	2	3	4
6	5	7	9	1	2	3	4
6	5	7	8	1	2	3	4
6	7	8	9	1	3	4	10
6	5	8	9	1	3	4	10
6	5	7	9	1	3	4	10
6	5	7	8	1	3	4	10
6	7	8	9	1	2	4	10
6	5	8	9	1	2	4	10
6	5	7	9	1	2	4	10
6	5	7	8	1	2	4	10

SOLUTION

6	7	8	9	1	2	4	4
6	5	7	8	1	2		4
6	7	8	9	1	3	4	10
6	5	7	8	1	3	4	10
6	7	8	9	1	2	4	10
6	5	7	8	1	2	4	10

```
 SOLUTION
6  7  8  9        1  2  3  4
6  5  7  8        1  2  3  4
6  7  8  9        1  2  4 10



6  5  7  8        1  2  4 10

 SOLUTION
6  7  8  9        1  2  3  4
6  5  7  8        1  2  3  4

 SOLUTION
6  7  8  9        1  2  3  4
6  5  7  8        1  2  3  4

 SOLUTION
6  7  8  9        1  2  3  4
6  5  7  8        1  2  3  4

 SOLUTION
6  7  8  9        1  2  3  4
6  5  7  8        1  2  3  4

 SOLUTION
6  5  7  8        1  2  3  4

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

 SOLUTION
6  5  7  8        1  2  3  4      10  9  0  0

**************************************************
```

Exemple B



TRANSITIONS

0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	0	1	0

SUCCESSEURS ET PREDECESSEURS

3	1	2	3	0	1	3	0	0	1	2	0	0	0	0	0	0	7
2	2	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	3
3	3	4	9	0	3	9	0	0	3	4	0	0	0	0	0	0	7
2	4	5	0	0	4	0	0	0	0	0	0	0	0	0	0	0	3
3	5	6	7	0	5	7	0	0	5	6	0	0	0	0	0	0	7
3	6	7	8	0	6	8	0	0	6	7	0	0	0	0	0	0	7
3	7	2	8	0	7	8	0	0	7	2	0	0	0	0	0	0	7
2	8	6	0	0	8	0	0	0	0	0	0	0	0	0	0	0	3
3	9	10	12	0	9	12	0	0	9	10	0	0	0	0	0	0	7
	10	2	11	0	10	11	0	0	10	2	0	0	0	0	0	0	7
	11	9	10	12	11	10	12	0	11	9	12	0	11	9	10	0	13
	12	11	0	0	12	0	0	0	0	0	0	0	0	0	0	0	2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3
4	2	1	7	10	2	7	10	0	2	1	10	0	2	1	7	0	13
2	3	1	0	0	3	0	0	0	0	0	0	0	0	0	0	0	3
2	4	3	0	0	4	0	0	0	0	0	0	0	0	0	0	0	3
2	5	4	0	0	5	0	0	0	0	0	0	0	0	0	0	0	3
3	6	5	8	0	6	8	0	0	6	5	0	0	0	0	0	0	7
3	7	5	6	0	7	6	0	0	7	5	0	0	0	0	0	0	7
3	8	6	7	0	8	7	0	0	8	5	0	0	0	0	0	0	7
3	9	3	11	0	9	11	0	0	9	3	0	0	0	0	0	0	7
3	10	9	11	0	10	11	0	0	10	9	0	0	0	0	0	0	7
3	11	10	12	0	11	12	0	0	11	10	0	0	0	0	0	0	7
3	12	9	11	0	12	11	0	0	12	9	0	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SOLUTION

11	9	10	12	2	1	7	[illegible]						
11	10	12	0	2	[illegible]	7	[illegible]						
11	9	10	0	2	[illegible]	7	[illegible]						

SOLUTION

1	9	10	12	2	1	7	3						
1	10	12	0	2	1	7	3						
1	9	10	0	2	1	7	3						
1	9	10	12	2	1	7	0						
1	10	12	0	2	1	7	0						
11	9	10	0	2	1	7	0						

SOLUTION

11	10	12	0	2	1	7	0	3	4	9	0		
11	10	12	0	2	1	7	0	3	9	0	0		
11	9	10	3	2	1	7	0						
11	9	10	12	2	1	7	0	3	4	0	0		
11	10	12	0	2	1	7	0	3	4	0	0		
11	9	10	0	2	1	7	0	3	4	0	0		

SOLUTION

11	10	12	0	2	1	7	5	3	4	9	0		
11	10	12	0	2	1	7	5	3	9	0	0		
11	9	10	3	2	1	7	5						
11	9	10	12	2	1	7	5	3	4	0	0		
11	10	12	0	2	1	7	5	3	4	0	0		
11	9	10	0	2	1	7	5	3	4	0	0		
11	10	12	0	2	1	7	0	3	4	9	0	5	6
11	10	12	0	2	1	7	0	3	9	5	6		
11	10	12	0	2	1	7	0	3	9	0	0	5	6
11	9	10	3	2	1	7	0	5	6	0	0		
11	9	10	12	2	1	7	0	3	4	5	6		
11	9	10	12	2	1	7	0	3	4	0	0	[illegible]	[illegible]
11	10	12	0	2	1	7	0	3	4	5	6		
11	10	12	0	2	1	7	0	3	4	0	0	5	[illegible]
11	9	10	0	2	1	7	0	3	4	5	6		
11	9	10	0	2	1	7	0	3	4	0	0	5	6

SOLUTION

11	10	12	0	2	1	7	5	3	4	9	0	6	[illegible]
11	10	12	0	2	1	7	5	3	9	6	8		
11	10	12	0	2	1	7	5	3	9	0	0	6	[illegible]
11	9	10	3	2	1	7	5	6	8	0	0		
11	9	10	12	2	1	7	5	3	4	6	8		
11	9	10	12	2	1	7	5	3	4	0	0	6	[illegible]
11	10	12	0	2	1	7	5	3	4	6	8		
11	10	12	0	2	1	7	5	3	4	0	0	6	[illegible]
11	9	10	0	2	1	7	5	3	4	6	8		
11	9	10	0	2	1	7	5	3	4	0	0	6	[illegible]
11	10	12	0	2	1	7	0	3	4	9	0	[illegible]	[illegible]
11	10	12	0	2	1	7	0	3	9	0	0	[illegible]	[illegible]
11	9	10	3	2	1	7	0	5	6	8	0		
11	9	10	12	2	1	7	0	3	4	0	0	[illegible]	8 0
11	10	12	0	2	1	7	0	3	4	0	0	[illegible]	8 0
11	9	10	0	2	1	7	0	3	4	0	0	[illegible]	8 0

SOLUTION

11	10	12	0	2	1	7	5	3	4	9	0	6	8	0	0
11	10	12	0	2	1	7	5	3	9	6	8				
11	10	12	0	2	1	7	5	3	9	0	0	6	8	0	0
11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0
11	10	12	0	2	1	7	5	3	4	6	8				
11	10	12	0	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	8	0	0
11	10	12	0	2	1	7	0	3	4	9	0	5	6	8	0
11	10	12	0	2	1	7	0	3	9	0	0	5	6	8	0
11	9	10	3	2	1	7	0	5	5	8	0				
11	9	10	12	2	1	7	0	3	4	0	0	5	6	8	0
11	10	12	0	2	1	7	0	3	4	0	0	5	6	8	0

11	9	10	0	2	1	7	0	3	4	0	0	5	6	8	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	12	2	1	7	0	3	4	0	0	5	6	8	0
11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	3	2	1	7	0	5	6	8	0				
11	9	10	0	2	1	7	0	3	4	0	0	5	6	8	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	12	2	1	7	0	3	4	0	0	5	6	8	0
11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	3	2	1	7	0	5	6	8	0				
11	9	10	0	2	1	7	0	3	4	0	0	5	6	8	0

SOLUTION

11	9	10	12	2	1	7	0	3	4	0	0	5	6	8	0
11	9	10	3	2	1	7	0	5	6	8	0				
11	9	10	0	2	1	7	0	3	4	0	0	5	6	8	0
11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	8	0	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	8	0	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	[illegible]	[illegible]		
11	9	10	12	2	1	7	5	3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
11	9	10	3	2	1	7	5	6	[illegible]	[illegible]	[illegible]		
11	9	10	0	2	1	7	5	3	[illegible]	[illegible]	[illegible]		
11	9	10	0	2	1	7	5	3	4	6	8	[illegible]	[illegible]	0	0

SOLUTION

11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	[illegible]	8	0	0
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	0	0	0

SOLUTION

11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	12	2	1	7	5	3	4	[illegible]	[illegible]				
11	9	10	12	2	1	7	5	3	4	0	0	[illegible]	[illegible]	0	0
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	[illegible]	[illegible]	0	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0
11	9	10	3	2	1	7	5	6	8	0	0				
11	9	10	0	2	1	7	5	3	4	6	8				
11	9	10	0	2	1	7	5	3	4	0	0	6	8	0	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	[illegible]	[illegible]

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	[illegible]	[illegible]

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0

SOLUTION

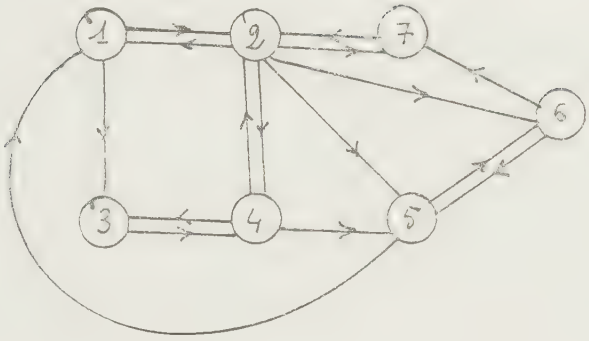
11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	8	0	0

SOLUTION

11	9	10	12	2	1	7	5	3	4	6	8				
11	9	10	12	2	1	7	5	3	4	0	0	6	[illegible]	0	0

```
  SOLUTION
11  9 10 12      2  1  7  5      3  4  6  8
11  9 10 12      2  1  7  5      3  4  0  0      6  8  0

  SOLUTION
11  9 10 12      2  1  7  5      3  4  6  8
11  9 10 12      2  1  7  5      3  4  0  0      6  8  0  0

  ***********************************************
```

Exemple C



TRANSITIONS

0	1	1	0	0	0	0
1	0	0	1	1	1	1
0	0	0	1	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	1	0
0	0	0	0	1	0	1
0	1	0	0	0	0	0

ETATS EQUIVALENTS : [illegible]

TRANSITIONS

0	1	1	0	0	0	0	[illegible]
1	0	0	1	0	0	0	0
0	0	0	1	0	0	0	[illegible]
0	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0
0	0	0	0	1	0	1	[illegible]
0	1	0	0	0	0	0	[illegible]
0	0	0	0	1	1	[illegible]	0

SUCCESSEURS ET PREDECESSEURS

3	1	2	3	0	1	3	0	0	1	2	0	0	0	0	0	0	7
3	2	1	4	0	2	4	0	0	2	1	0	0	0	0	0	0	7
2	3	4	0	0	3	0	0	0	0	0	0	0	0	0	0	0	3
4	4	2	3	5	4	3	5	0	4	2	5	0	4	2	3	0	13
3	5	1	6	0	5	6	0	0	5	1	0	0	0	0	0	0	7
3	6	5	7	0	6	7	0	0	6	5	0	0	0	0	0	0	7
2	7	2	0	0	7	0	0	0	0	0	0	0	0	0	0	0	3
4	8	5	6	7	8	6	7	0	8	5	7	0	8	5	6	0	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	2	5	0	1	5	0	0	1	2	0	0	0	0	0	0	7
4	2	1	4	7	2	4	7	0	2	1	7	0	2	1	4	0	13
3	3	1	4	0	3	4	0	0	3	1	0	0	0	0	0	0	7
3	4	2	3	0	4	3	0	0	4	2	0	0	0	0	0	0	7
4	5	4	6	8	5	6	8	0	5	4	8	0	5	4	6	0	13
3	6	5	8	0	6	8	0	0	6	5	0	0	0	0	0	0	7
3	7	6	8	0	7	8	0	0	7	6	0	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SOLUTION

4	2	3	5	8	6	7	0
4	3	5	0	8	6	7	0
4	2	5	0	8	6	7	0

SOLUTION

4	2	5	1	8	6	7	0

LGNE A DEDOUBLER. 25

ETATS EQUIVALENTS : 5 = 9

TRANSITIONS

0	1	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	1	0	0

0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

SUCCESSEURS ET PREDECESSEURS

3	1	2	3	0	1	3	0	0	1	2	0	0	0	0	0	0	7
3	2	1	4	0	2	4	0	0	2	1	0	0	0	0	0	0	7
2	3	4	0	0	3	0	0	0	0	0	0	0	0	0	0	0	3
4	4	2	3	9	4	3	9	0	4	2	9	0	4	2	3	0	13
3	5	1	6	0	5	6	0	0	5	1	0	0	0	0	0	0	7
3	6	5	7	0	6	7	0	0	6	5	0	0	0	0	0	0	7
2	7	2	0	0	7	0	0	0	0	0	0	0	0	0	0	0	3
4	8	6	7	9	8	7	9	0	8	6	9	0	8	6	7	0	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	2	5	0	1	5	0	0	1	2	0	0	0	0	0	0	7
4	2	1	4	7	2	4	7	0	2	1	7	0	2	1	4	0	13
3	3	1	4	0	3	4	0	0	3	1	0	0	0	0	0	0	7
3	4	2	3	0	4	3	0	0	4	2	0	0	0	0	0	0	7
2	5	6	0	0	5	0	0	0	0	0	0	0	0	0	0	0	3
3	6	5	8	0	6	8	0	0	6	5	0	0	0	0	0	0	7
3	7	6	8	0	7	8	0	0	7	6	0	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	9	4	8	0	9	8	0	0	9	4	0	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SOLUTION

4	2	3	9	8	6	7	0
4	3	9	0	8	6	7	0
4	2	9	0	8	6	7	0

SOLUTION

4	2	9	1	8	6	7	0

SOLUTION

4	2	9	1	8	6	7	0

SOLUTION

4	2	9	1	8	6	7	0

SOLUTION

4	2	9	1	8	6	7	5

SOLUTION

4	2	9	1	8	6	7	5

SOLUTION

4	2	9	1	8	6	7	5

LGNE A DEDOUBLER. 23

ETATS EQUIVALENTS : 3 = 10

TRANSITIONS

0	1	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	1	0
1	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

SUCCESSEURS ET PREDECESSEURS

3	1	2	10	0	1	10	0	0	1	2	0	0	0	0	0	0			1
3	2	1	4	0	2	4	0	0	2	1	0	0	0	0	0	0			7
2	3	4	0	0	3	0	0	0	0	0	0	0	0	0	0	0			3
4	4	2	3	9	4	3	9	0	4	2	9	0	4	2	9	0			13
3	5	1	6	0	5	6	0	0	5	1	0	0	0	0	0	0			
3	6	5	7	0	6	7	0	0	6	5	0	0	0	0	0	0			
2	7	2	0	0	7	0	0	0	0	0	0	0	0	0	0	0			
4	8	6	7	9	8	7	9	0	8	6	9	0	8	6	9	0			13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
3	1	2	5	0	1	5	0	0	1	2	0	0	0	0	0	0			7
4	2	1	4	7	2	4	7	0	2	1	7	0	2	1	4	0			13
2	3	4	0	0	3	0	0	0	0	0	0	0	0	0	0	0			
3	4	2	3	0	4	3	0	0	4	2	0	0	0	0	0	0			7
2	5	6	0	0	5	0	0	0	0	0	0	0	0	0	0	0			3
3	6	5	8	0	6	8	0	0	6	5	0	0	0	0	0	0			7
3	7	6	8	0	7	8	0	0	7	6	0	0	0	0	0	0			7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			1
3	9	4	8	0	9	8	0	0	9	4	0	0	0	0	0	0			
2	10	1	0	0	10	0	0	0	0	0	0	0	0	0	0	0			1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0

```
 SOLUTION
4  2  3  9       8  6  7  0
4  3  9  0       8  6  7  0
4  2  9  0       8  6  7  0

 SOLUTION
4  2  9  1       8  6  7  0

 SOLUTION
4  2  9  1       8  6  7  0

 SOLUTION
4  2  9  1       8  6  7  0

 SOLUTION
4  2  9  1       8  6  7  5

 SOLUTION
4  2  9  1       8  6  7  5



 SOLUTION
4  2  9  1       8  6  7  5

 SOLUTION
4  2  9  1       8  6  7  5

 SOLUTION
4  2  9  1       8  6  7  5

 SOLUTION
4  2  9  1       8  6  7  5

 SOLUTION
4  2  9  1       8  6  7  5

 SOLUTION
4  2  9  1       8  6  7  5       3  0  0  0

 SOLUTION
4  2  9  1       8  6  7  5       3  0  0  0

 SOLUTION
4  2  9  1       8  6  7  5       3  0  0  0

 SOLUTION
4  2  9  1       8  6  7  5       3  0  0  0

 SOLUTION
4  2  9  1       8  6  7  5       3 10  0  0
4  2  9  1       8  6  7  5       3  0  0  0      10  0  0  0
*STOP* 0@
JOB STEP 03 TERMINATED AFTER 0000.57MIN
.        CORE USED  0050      DISC USED  0017    WAIT TIME 0000.23
*****************************************************************
```

B I B L I O G R A P H I E

[1] HARTMANIS J, STEARNS R.-E. - Algbraic Structure of sequential machines - Prentice Hall, Inc. 1966

KINNEY H. - Decomposition of asynchronous sequential suitching circuits - IEEE Tran. on Comp., Vol. C19 n°6, Juin 1970.

KUZNETSOV O.-P. - The parallel decomposition of automata with separation of inputs - Automatika i Telemekhanika n°3 pp. 104-109, Mars 1969.

[2] GIRARD P., NASLIN P. - Construction des machines séquentielles industrielles - Dunod 1973.

[3] DAVID R. - Réalisation de systèmes séquentiels asynchrones par interconnexion, simple de cellules séquentielles identiques - Thèse de doctorat d'Etat - Université de Grenoble, 1969.

DAVID R., PERRET R. - "Principe d'une cellule élémentaire séquentielle" C.R.Ac.Sc. de Paris t. 262, pp. 656-58, 1966.

[4] ZADEH L.-A., POLAK E. - System theory - Mc Graw-Hill 1969.

ARBIB M.-A. - Theory of abstract automata - Prentice-Hall.

[5] GLUSHKOV V.-M. - Some synthesis problems of digital automata Zh. Vychislit. Matemat. i Matem. Fiz., Vol. N° 3, Mai-Juin 1961.

[6] BOOTH T.-L. - Sequential machines and Automata theory
John Viley and Sons, Inc. 1967.

AISERMAN, GUSEV, ROZONOER, SMIRNOVA, TAL'
Logic, Automata, and Algorithms - Academis Press, Inc. 1971.

[7] GINZBURG - Algebraic theory of Automata -
Academic Press, Inc. 1968.

[8] BERGE G- Graphes et hypergraphes - Dunod 1971.

NOM DE L'ETUDIANT : BENLAHCEN Driss

NATURE DE LA THESE : DOCTEUR INGENIEUR

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 18 Juin 1975

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

UNIVERSITÉ DE NANCY I
Le Président

J.R. HELLUY